File Number S360(Mod.20)-33
Form C26-3806-1

IBM Systems Reference Library

IBM System/360 Model 20 Disk Programming System Disk Sort/Merge Program

This publication describes the functions and the use of the Disk Sort/Merge Program for the IBM System/360 Model 20. It contains the following information:

1. Minimum machine requirements for sorting or merging records with this program.
2. Additional machine features supported.
3. Program capabilities.
4. A description of the control statements required to define sort or merge operations.
5. A description of the facilities provided for inserting user-written routines into the program.

For a list of associated publications and their abstracts, see IBM System/360 Model 20 Bibliography, Form A26-3565.

PREFACE

This manual describes the IBM System/360 Model 20, Disk Programming System (DPS), Disk Sort/Merge Program. It includes a description of the program, the machine configurations to which the program applies, and the format of the control statements needed to tailor the Disk Sort/Merge Program to a specific job.

Amplified details, technical data, procedures for program modification, and background information are also included.

The program operates in conjunction with the Model 20 DPS Control and Service programs. Therefore, in addition to a working knowledge of the System/360 Model 20, the reader should be familiar with the publication IBM System/360 Model 20, Disk Programming System, Control and Service Programs, Form C24-9006. Certain Job Control statement information has been incorporated into this manual to minimize the need to refer to other publications.

If user exits are used, the reader should be familiar with the SRL publication IBM System/360 Model 20, Disk and Tape Programming Systems, Assembler Language, Form C24-9002.

A glossary is appended to explain terminology used in this publication that may be unfamiliar to the reader.

Second Edition

This is a major revision of, and obsoletes, C26-3806-0 and TNL N24-9014. The publication has been completely rewritten and new material has been incorporated. Therefore, this edition should be reviewed in its entirety.

Specifications contained herein are subject to change from time to time. Any such changes will be reported in subsequent revisions or Technical Newsletters.

This publication was prepared for production using an IBM computer to update the text and to control the page and line format. Page impressions for photo-offset printing were obtained from an IBM 1403 printer using a special print chain.

Requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form is provided at the back of this publication for readers' comments. If the form has been removed, comments may be addressed to IBM Laboratories, Programming Publications, 703 Boeblingen/Germany, P. O. Box 210.

Many data processing applications require that certain data files be sorted into some predetermined sequence. Once in the proper sequence, files of the same format must often be merged together. If the task of sorting data records and merging data files were generalized so that a single program could be used for most sequencing and merging tasks, much repetitive programming could be avoided.

The IBM System/360 Model 20, Disk Programming System (DPS), Disk Sort/Merge Program, referred to in this manual as "the Disk Sort/Merge Program", allows a user to sort or merge files having a wide variety of characteristics by simply preparing control statement cards which include a definition of the data to be processed (called a file) and a description of the processing to be done, thus freeing the user to concentrate his programming efforts on other tasks in his data processing application.

The Disk Sort/Merge Program can be card-resident or disk-resident.

PROGRAM FEATURES

The IBM System/360 Model 20 Disk Sort/Merge Program processes fixed-length records, blocked or unblocked, and provides the user with the ability to

- Sort a single data file, containing data in random or unknown sequence or order, and arrange the logical records according to a user-specified sequence, taking advantage of any sequencing which already exists in the input file. (See Figure 1A.)
- Merge from two to four already sequenced input files into a single ordered file, referred to as merging with an order of two, three, or four depending on the number of input files used. (See Figure 1B).
- Sequence check a single file while copying it, with optional reblocking, referred to as merging with an order of one.

Sorting or merging takes place on IBM 1316 Disk Packs contained on one or two IBM 2311 Disk Storage Drives (Model 11 or 12).

Output can be to IBM 1316 Disk Packs, or magnetic tape, cards, or to a combination of either disk and card or magnetic tape and card. In the latter two cases the entire output file is written twice.

Input can be from IBM 1316 Disk Packs, magnetic tape, or card. For sorting, input can also be from a combination of disk and card or magnetic tape and card, where each of the two units contains part of the file. For merging, one of the input files can be from a card reader with the other input files from either disk or magnetic tape, but not both.

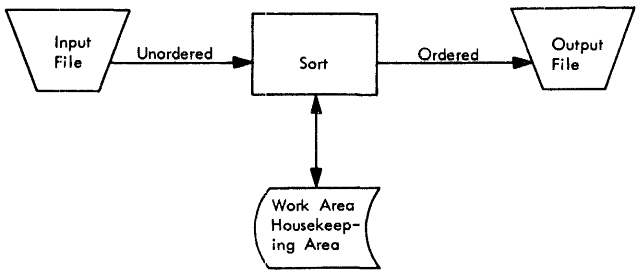


Figure 1A. Sorting Process

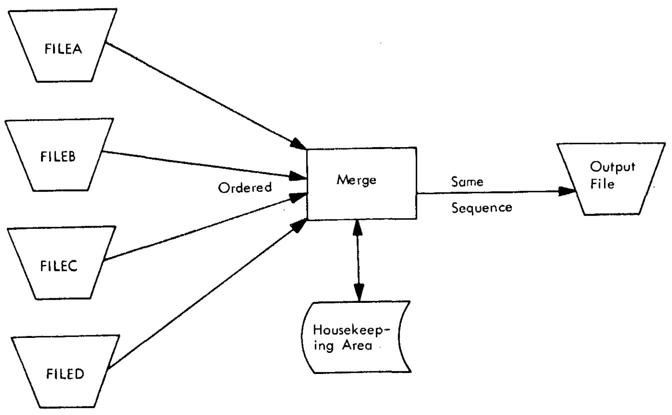


Figure 1B. Merging Process

Figure 1C symbolically represents the permissible I/O configurations.

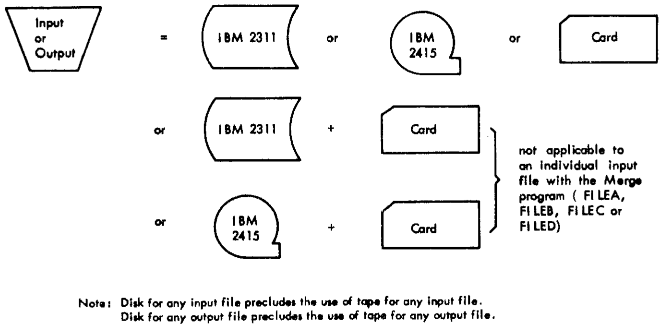

Input or Output
=
IBM 2311
or
IBM 2415
or
Card
or
IBM 2311
+
Card
or
IBM 2415
+
Card
not applicable to an individual input file with the Merge program (FILEA, FILEB, FILEC or FILED)
Note: Disk for any input file precludes the use of tape for any input file.
Disk for any output file precludes the use of tape for any output file.

Figure 1C. Input/Output Configurations

The Disk Sort/Merge Program is a generalized program; that is, it is designed to satisfy a wide variety of application requirements. Each time a specific operation is to be performed, certain items of information pertaining to files, records, available machine components, and the options which the user wishes to invoke must be supplied. This information is supplied by control statements which are punched into cards and read at execution time. This information is used in the first phase to tailor the generalized sorting or merging functions to the application specified by the user.

The Disk Sort/Merge Program uses information in each logical record to determine the position of the record in the ordered output file. The parts of the logical records containing information used for this purpose are called control fields. (See Control Fields and Collating Sequence.) The user may specify up to twelve control fields. Both the control fields and the remaining fields of the logical record may contain information the user considers his "data". For example, an item-code entry could be used as a control field and then as data in another program. The division of each logical record into control fields and data fields does not affect subsequent use of the data, since the logical records are not altered.

There are two types of control statements used. The first type, Job Control statements, is read by the IBM System/360 Model 20 DPS control program. These statements describe the system to be used and the labels which are to be processed. The second type, Sort/Merge Control statements, describe the actual sorting or merging job to be performed. These statements are read by the Disk Sort/Merge Program.

The Disk Sort/Merge Program includes the following specific features:

1. Uses mnemonic control statement information that describes input files, output files, sequence of sorting or merging, etc.

2. Sorts or merges on one to twelve control fields containing a total of not more than 256 bytes of control data.

3. Permits the specification of ascending or descending sequence for each control field.

4. Sorts or merges on control fields with data in binary, alphameric-character, zoned-decimal, packed-decimal, or fixed-point integer format. However, for any one sort/merge application, all control fields must have the same format.

5. Provides the option of writing an output file composed of the sorted disk addresses of the records instead of the records themselves.

6. Provides the option of deleting records containing a specified character in a specified position of the record (called the blank record indicator).

7. Provides the option of bypassing unreadable tape or disk data blocks.

8. Provides optional checkpoint, interrupt, and restart procedures to be used with sort operations.

9. Provides exits from the Disk Sort/Merge Program to user-written routines.

10. Processes standard System/360 Volume and File Labels.

11. Provides for the specification of an alternate input tape drive (sort operations only) and an alternate output tape drive (sort or merge operations).

12. Provides for multi-volume tape input/output files.

13. Provides for multi-extent and multi-pack disk input/output files; however, all packs for a file must be read or written from the same drive.

14. Provides the option of printing control statement information and intermediate pass-count messages.

15. Provides printed diagnostics and end-of-phase messages.

16. Provides for either card-residence or disk-residence of the Disk Sort/Merge Program.

SYSTEM CONFIGURATION

The System/360 used to run the Model 20 DPS Disk Sort/Merge Program must have a minimum of:

- IBM 2020 Processing Unit Model BC2 (12,288 bytes of core storage)
- One IBM 2311 Disk Storage Drive, Model 11 or 12, with IBM 1316 Disk Pack
- One IBM 1403 (Model 2, 7, or N1) or 2203 Printer
- One IBM 2501, 2560, or 2520 Model A1 Card Reader

The Disk Sort/Merge Program also supports the following:

- Maximum of 16,384 bytes of core storage
- Maximum of two IBM 2311 Disk Storage Drives, Model 11 or 12
- One to four IBM 2415 Magnetic Tape Drives (7 and/or 9 track) for input/output to a sort
- One to six IBM 2415 Magnetic Tape Drives (7 and/or 9 track) for input/output to a merge
- One IBM 1442 Card Punch or 2520 Card Read-Punch or 2560 MFCM.

A control field is a group of contiguous bytes within a record which, in effect, is compared with the corresponding field of every record in a file to determine the desired sequential position of that record within the output file. The Disk Sort/Merge Program is capable of sorting or merging records on a maximum of twelve control fields, with a maximum total length of 256 bytes.

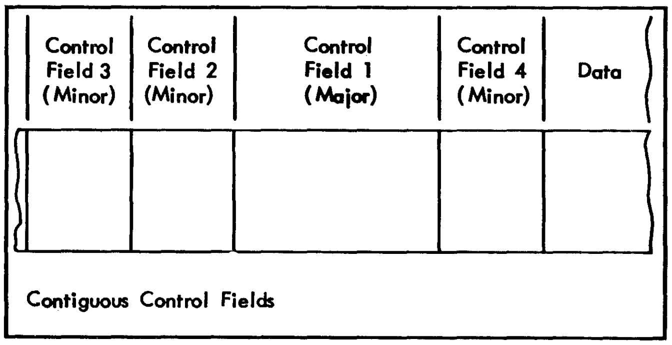
The most significant field is the major control field. Corresponding major control fields in two records are compared first and, if they compare unequally, no further control fields are compared between the two records. The other fields may be regarded as minor. They will be compared according to the relative precedence which they are assigned by the user. The individual fields may be contiguous or separated and may be located anywhere within a record; however, a particular control field must be located in the same relative position in each record of a file. Figure 2 shows some Control Field Arrangements. Control fields may overlap if binary (BI) or alphameric-character (CH) format is used.

Control data must consist of one of the following:

1. Alphameric-character data (CH)
2. Binary data (BI)
3. Packed-decimal data (PD)
4. Zoned-decimal data (ZD)
5. Fixed-point integer data (FI)

The following limitations must be observed when planning the characteristics of the control fields for a file:

1. Each control field must begin and end on a byte boundary.
2. Each control field must be at least one byte long.
3. Each control field can be a maximum of:
   a. 256 bytes for binary, alphameric-character, or fixed-point integer fields, or
   b. 16 bytes, including sign, for packed-decimal and zoned-decimal fields.
4. If disk-address output is specified, the total length of all control fields must be less than the input record length by at least 8 bytes.
5. Only one format can be specified for all control fields.
6. No overlapping control fields should be specified for zoned-decimal, packed-decimal or fixed-point integer formats.



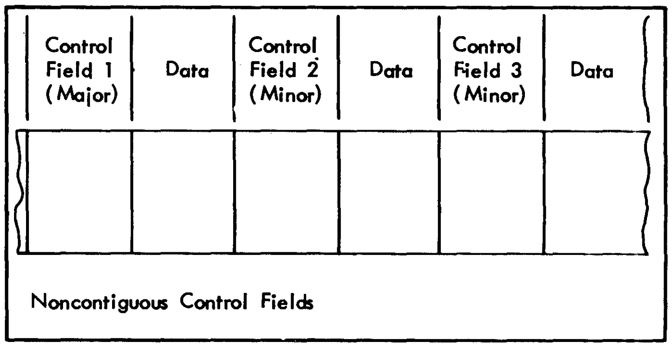


Figure 2. Some Control Field Arrangements

Files may be sorted or merged into either ascending or descending sequence for each control field. Therefore, each control field must have its sequence specified individually. For example, with three control fields a user might specify ascending sequence for the major field, descending sequence for the first minor field, and ascending sequence for the second minor field. In the output file all groups of records which are equal on the major field are placed into descending sequence on the first minor field, and within groups which are equal on both the major and first minor control fields, the records are placed into ascending sequence on the last minor field.

The control fields are defined by user-prepared control statements. These statements specify the type of operation (sort or merge), the sequence of each control field (ascending or descending), the size, and the location within a record of each field. (See Sort/Merge Control Statements.)

As an example, consider the following file:

	Control Fields		
	Major	Minor 1	Minor 2
Record 1	72	83	13
Record 2	36	10	67
Record 3	72	83	12
Record 4	15	89	47
Record 5	36	19	02

The file is to be sorted into ascending (increasing) order. Data in the major field is compared first and the records are then in the following order: 4, 2 and 5, 1 and 3. Where the major fields are equal (2 and 5, 1 and 3), the first minor field is considered. Since 10 precedes 19, record 2 precedes record 5. But records 1 and 3 have equal first minor fields. The second minor field is then tested and, since 12 is less than 13, record 3 precedes record 1. The final sorted file will be:

	Control Fields		
	Major	Minor 1	Minor 2
Record 4	15	89	47
Record 2	36	10	67
Record 5	36	19	02
Record 3	72	83	12
Record 1	72	83	13

If the sequence had been specified as ascending for the major field and the second minor field, but as descending (decreasing) for the first minor field, then record 5 would precede record 2. The final order would then be: Record 4, Record 5, Record 2, Record 3, Record 1.

Collating sequence is binary (00000000 to 11111111). The Disk Sort/Merge Program includes data conversion routines to convert fixed-point integer (FI), packed-decimal (PD), and zoned-decimal (ZD) input records to binary format and then to reconvert them before the final output file is written. When using either zoned-decimal data or packed-decimal data, the signed fields must be either all ASCII or all EBCDIC; they must not contain a combination of both. Binary (BI) and alphameric-character (CH) format need not be translated since the EBCDIC collating sequence presumes a binary comparison. (See SORT Statement for further details.)

Figure 3 shows the entire set of 256 punch combinations, each of which can be recognized by the System/360 as a unique character, referred to as the Extended Binary-Coded Decimal Interchange Code (EBCDIC).

Each of the 256 possible characters is made up of two half-bytes -- the "upper" and the "lower" half-byte -- each containing four bits. Since $2^4 = 16$, this yields 16 upper half-bytes, each of which is associated with one of 16 lower half-bytes to yield a 16 x 16 (= 256) hexadecimal-code table.

Each of the characters (0-9, A-F) printed horizontally across the top of the hexadecimal table represents a different internal configuration of four bits for the upper half-byte. Similarly, the vertically ordered marginal characters (also 0-9, A-F) designate the bits of the lower half-bytes.

Each box at the intersection of a row and a column stands for a different byte, formed in core storage by the bits of the upper and the lower half-bytes represented by the arbitrary column and row labels.

For example: The digit 2 is formed in core storage by the four bits represented by the upper half-byte labelled F, and the four bits represented by the lower half-byte labelled 2.

The codes shown above the diagonal line of each of the 256 boxes in the table are the card punch combinations that correspond to the 256 hexadecimal codes. (The letters T, E, and Z have been used to represent 12-, 11-, and zero-punches, respectively.)

The symbol below the diagonal line in some of the boxes is the character or graphic assigned to the punch combination.

The standard ascending collating sequence extends through all rows of one column before preceding to the next column, i.e., column 0, row 0; column 0, row 1;...; column 0, row F; column 1, row 0;...; column F, row F.

EBCDIC - THE FOUR HIGH-ORDER BITS

	BITS	FIRST QUADRANT				SECOND QUADRANT				THIRD QUADRANT				FOURTH QUADRANT			
		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
HEXADECIMAL CODE		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0000	TZ981 NULL	TE981	EZ981	TEZ981	BLANK	T &	E -	TEZ	TZ81	TE81	EZ81	TEZ81	TZ	EZ	Z82	Z 0
1	0001	T91	E91	Z91	91	TZ91	TE91	Z1 /	TEZ91	TZ1 a	TE1 j	EZ1	TEZ1	T1 A	E1 J	EZ91	1 1
2	0010	T92	E92	Z92	92	TZ92	TE92	EZ92	TEZ92	TZ2 b	TE2 k	EZ2 s	TEZ2	T2 B	E2 K	Z2 S	2 2
3	0011	T93	E93	Z93	93	TZ93	TE93	EZ93	TEZ93	TZ3 c	TE3 l	EZ3 t	TEZ3	T3 C	E3 L	Z3 T	3 3
4	0100	T94	E94	Z94	94	TZ94	TE94	EZ94	TEZ94	TZ4 d	TE4 m	EZ4 u	TEZ4	T4 D	E4 M	Z4 U	4 4
5	0101	T95	E95	Z95	95	TZ95	TE95	EZ95	TEZ95	TZ5 e	TE5 n	EZ5 v	TEZ5	T5 E	E5 N	Z5 V	5 5
6	0110	T96	E96	Z96	96	TZ96	TE96	EZ96	TEZ96	TZ6 f	TE6 o	EZ6 w	TEZ6	T6 F	E6 O	Z6 W	6 6
7	0111	T97	E97	Z97	97	TZ97	TE97	EZ97	TEZ97	TZ7 g	TE7 p	EZ7 x	TEZ7	T7 G	E7 P	Z7 X	7 7
8	1000	T98	E98	Z98	98	TZ98	TE98	EZ98	TEZ98	TZ8 h	TE8 q	EZ8 y	TEZ8	T8 H	E8 Q	Z8 Y	8 8
9	1001	T981	E981	Z981	981	T81	E81	Z81	81	TZ9 i	TE9 r	EZ9 z	TEZ9	T9 I	E9 R	Z9 Z	9 9
A	1010	T982	E982	Z982	982	T82 ¢	E82 !	TE	82 :	TZ82	TE82	EZ82	TEZ82	TZ982	TE982	EZ982	TEZ982
B	1011	T983	E983	Z983	983	T83 .	E83 $	Z83 ,	83 #	TZ83	TE83	EZ83	TEZ83	TZ983	TE983	EZ983	TEZ983
C	1100	T984	E984	Z984	984	T84 ⌑ or <	E84 *	Z84 %	84 @	TZ84	TE84	EZ84	TEZ84	TZ984	TE934	EZ984	TEZ984
D	1101	T985	E985	Z985	985	T85 (	E85)	Z85 _	85 '	TZ85	TE85	EZ85	TEZ85	TZ985	TE985	EZ985	TEZ985
E	1110	T986	E986	Z986	986	T86 +	E86 ;	Z86 >	86 =	TZ86	TE86	EZ86	TEZ86	TZ986	TE986	EZ986	TEZ986
F	1111	T987	E987	Z987	987	T87 |	E87 ¬	Z87 ?	87 "	TZ87	TE87	EZ87	TEZ87	TZ987	TE987	EZ987	TEZ987

EBCDIC - THE FOUR LOW-ORDER BITS

HEXADECIMAL CODE T = 12 - PUNCH E = 11 - PUNCH Z = ZERO-PUNCH

Figure 3. Code Structure - Hexadecimal Code Table

Hence, for example.

1. The period symbol (.) precedes (i.e., is lower than) the exclamation mark (!).

2. The exclamation mark (!) precedes the 11-punch (-).

3. All assigned special characters precede the alphabet.

4. The alphabet precedes (i.e., is lower in sequence than) the digits.

5. Punch combination E987 precedes the combination EZ981.

Thus, the standard ascending collating sequence for the most commonly used characters is:

blank, &, -, /, A-Z, 0-9.

(The converse for standard descending sequence.)

No graphics appear in the hexadecimal-code table (Figure 3) for the punch combinations TZ and EZ, because they are not part of the defined BCDIC subset. These combinations occur when a numeric input field is signed and the low-order position of the field happens to contain a zero. They can also occur in other than the low-order position if cards with zone overpunches in numeric fields are read in the input file.

Figure 4 contains the standard EBCDIC collating sequence for 63 common assigned graphics.

Ascending	ƀ (blank)	A	1	
Sequence:	¢	B	2	
↓	.	C	3	
	⌑ or <	D	4	
	(	E	5	
	+	F	6	
	|	G	7	
	&	H	8	
	!	I	9	↑
	$	J		
	*	K		
	)	L		
	;	M		Descending
	¬	N		Sequence:
	-	O		
	/	P		
	,	Q		
	%	R		
	_	S		
	>	T		
	?	U		
	:	V		
	#	W		
	@	X		
	'	Y		
	=	Z		
	"	0		

Figure 4. Standard EBCDIC Collating Sequence for 63 Common Assigned Graphics

The user has the option of designating a unique collating sequence (ASCII, for example). By using a programmed exit and his own translation routine, he can sort or merge his records in accordance with any collating sequence he wishes to develop. (See Exit 12, Exit 32, Exit 42, and Exit 44.)

The Disk Sort/Merge Program can sort or merge fixed-length tape or disk records or sort unblocked card records with a maximum length of 80 characters.

The fixed-length tape records may be unblocked, in fixed-length blocks, or (for input only) in variable-length blocks.

Variable-length records can be read only if a user-written routine is added to convert the variable-length records to fixed-length records (See Exit 12).

Input is accepted in contiguous blocks of the specified block length. Thus, disk input may include sequential or direct-access files but not indexed-sequential files.

Certain records may be automatically deleted by designating that records with a specified character (blank or any other character may be used) in a specified position of a record are to be deleted. (See DELBLANK, described under RECORD Statement.) This is of special use with direct-access files.

The first record of a block normally is assumed to begin in the first position of that block. An option is provided which allows the first record of an input block to begin in other than the first position of that block so that some or all of the unused bytes (which could be used for chain addresses, identification, etc.) can be left at the start of the block rather than the end. (See SKIPBYTE, described under INPFIL Statement.)

Regardless of block length, the record length must not exceed 4096 bytes. All records must be at least 3 bytes long.

For disk records, the specified block lengths will be expanded to the next larger multiple of 270 bytes unless the specified block length is itself a multiple of 270 bytes. There will then be several unused bytes in each disk block. These characters are ignored on input. Each block starts on a sector boundary. The Disk Sort/Merge Program automatically takes care that no block crosses a cylinder boundary.

Note: Direct-access files allow a block or blocks to cross cylinder boundaries. If a block crosses a cylinder boundary, a merge or sort operation is not possible.

The maximum input/output block lengths are dependent upon the amount of core storage available, the number of optional functions specified, and the amount of core storage, if any, taken up by user routines. The block lengths are also dependent upon the sort technique used internally or the order of merge in a merge application. The two sort techniques are described under Program Description and are called the Uni-Resident sort technique (UR) and the Variable-String Interleave sort technique (VSI).

An end-of-file condition is recognized by a /*b in the first three bytes of a record, or by having reached the end of the input extent. End-of-file is also recognized if the end of the final extent is reached.

If the optional functions are to a minimum, the maximum block lengths in Figure 5 apply. The block lengths shown in Figure 5 for disk are specified block sizes; the actual block sizes read or written are always multiples of 270 bytes so that entire sectors are read or written at one time. If the user-supplied block length is too large (e.g., input block size of 6750 is given with 16,384 bytes of core storage and options are used which restrict the maximum input block size to a smaller number), then an appropriate diagnostic message is printed.

No end-of-file record is written for disk output if the output is limited by the specified extent. The maximum and minimum record lengths may not be possible when used with the maximum or minimum block lengths because for extremely small record lengths, the maximum block length will be shortened to some degree. If the maximum record length for a particular core storage size is used, there can be only one record per block. For block lengths and/or record lengths other than the maximums, there are two methods of determining whether or not a given combination of record and block lengths can be used. The first method is to go through the computations in IBM System/360 Model 20, Disk Programming System, Performance Estimates, Form C33-6003. The other method is to use the timing program (phase 5) to test the desired record and block length combination. If the combination is not allowed, an appropriate diagnostic message will be printed.

	Block Size				Record Length			
	Disk		Tape		Disk		Tape	
Core Storage	12288	16384	12288	16384	12288	16384	12288	16384
Maximum Length:								
Sort (UR)	5400	6750	4095	4095	1347	2697	1347	2697
Sort (VSI)	5400	6750	4095	4095	1350	2430	1350	2430
Merge (order = 1)	2700	6750	4095	4095	2700	4096	4395	4095
Merge (order = 2)	2700	2700	3133	4095	2700	2700	3133	4095
Merge (order = 3)	1350	2700	2350	3375	1350	2700	2350	3375
Merge (order = 4)	1350	2700	1880	2700	1350	2700	1880	2700
Minimum (Sort or Merge)	1 Sector	1 Sector	18	18	3	3	18	18

Figure 5. Block Size and Record Lengths in Bytes for Input and Output

For merging, the maximum record lengths and block lengths listed in Figure 5 may exist in combination with each other. Tape records cannot be less than 18 bytes long.

Card input/output files with fixed-length format can be processed. Records must be unblocked and can be a maximum of 80 characters in length. An EOF (end-of-file) card -- /*b in columns 1, 2, and 3 -- is required at the end of an input card file. Column binary data is not allowed. Card input/output files are considered to be unlabelled.

For sorting, the input file can consist of a combination of disk and card or magnetic tape and card. For merging, one of the input files can be a card file with the other input files from either disk or magnetic tape, but not both. (See Figure 1C.)

The checkpoint (CKPT) and RESTART options permit the user to interrupt the sorting process during phase 2 or phase 3 and restart it at the last checkpoint record written. Refer to the section Program Description for an explanation of program phases. This feature can be used only during sort runs and only if two work areas and CKPT have been specified. The checkpoint option is specified in the SORT statement. The Disk Sort/Merge Program writes a checkpoint record into the housekeeping area during each pass of phase 2. The message "FIRST CHECKPOINT" is printed at the beginning of phase 2. The checkpoint records contain the information required to restore the Disk Sort/Merge Program to the beginning of the interrupted pass. Processing can be continued from this point.

No checkpoints are written during phase 1. Any sort operation that is interrupted before the beginning of phase 2 must be restarted from the beginning. If the Disk Sort/Merge Program is interrupted during phase 2 or phase 3, the user may restart from the last checkpoint passed prior to interruption. To do so, he must remove the OPTION statement card, punch RESTART as an additional operand, and reinsert the card in the deck. Refer to the section OPTION Statement. The Disk Sort/Merge Program and all control-statement cards are then loaded as if the run were being restarted from the beginning; however, all sorting that has taken place prior to the last checkpoint will be bypassed. RESTART may only be specified after an interrupt has occured.

Note: Each disk pack must be returned to the drive it was mounted on at the time the interrupt occurred; and the disk storage areas used at restart time must contain the data present when the interruption occurred.

RECORD COUNTING

The record-count printout must be specified by the user. Placing the operand PRINT in the OPTION statement (see OPTION Statement) will cause a count of the records processed to be printed at the end of phase 1, phase 3, and phase 4. The number of records deleted by the DELBLANK option are printed at the end of phase 1. The number of records deleted and/or inserted by the user program are printed at the end of phase 3.

A sequence check is performed during phase 4. If a sequence error is detected, a message is printed and a halt occurs.

Only one error per card statement per pass will be detected. If other errors are present, they will only be detected after the first errors have been corrected.

Errors in Job Control statements are detected by the Job Control Program, and diagnostic messages are printed.

Diagnostics are included for all Sort/Merge control statements and diagnostic messages are printed. See Appendix D for Diagnostic Messages.

For input and output, the error recovery procedures specified for DPS IOCS are used.

In order to provide certain essential label information for processing standard labels, and to provide for unit assignments, the Disk Sort/Merge Program utilizes a subset of the standard System/360 Job Control statements. For complete details on these statements, see IBM System/360 Model 20, Disk Programming System Control and Service Programs, Form C24-9006.

The following eight general Job Control statements can be used:

JOB
ASSGN
DATE
CONFG
LOG
NOLOG
FILES
EXEC

The following four file Job Control statements are used for processing labelled output files, input files, work areas, and housekeeping area:

VOL
TPLAB
DLAB
XTENT

JOB CONTROL STATEMENT FORMAT

All Job Control statements use the following format:

The Name consists of "//" in columns 1 and 2 and it must be followed by at least one blank.

The Operation Code follows the Name to describe the type of Job Control statement and must be one of the twelve statements listed above. It must be followed by at least one blank.

The Operand list follows the Operation Code. It may be blank or contain several operands and values, in the format prescribed for the particular statement type, and separated by commas. The operand list must be followed by at least one blank, and it cannot extend beyond column 71 except that the terminating blank can be in column 72.

Continuation of a statement into the following card is denoted by a non-blank character in column 72. A card following a card with a non-blank column 72 must have columns 1-15 blank, with information starting in column 16. If column 72 is non-blank, the last entry on the card before column 72 must be a comma. The only Job Control statement which can be continued is the DLAB statement, which must be divided as shown under DLAB Statement.

User Job Identification may be given in columns 73-80. It is not checked by the Disk Sort/Merge Program.

GENERAL JOB CONTROL STATEMENTS

JOB Statement

The JOB statement must be the first card of the Control statement deck except for LOG or NOLOG. The following format is used for both sorting and merging applications:

```
// JOB SORT
```

ASSGN Statement

An array of symbolic input and output devices is assumed to be available for accomplishing the sort or merge operation defined by the user. At execution time the physical identity of each symbolic device must be identified by Assign (ASSGN) statements. One ASSGN statement must be prepared for each symbolic unit used, except that the ASSGN statement can be omitted if the information on it is still in core storage from a previous job.

The basic format of the ASSGN statement is:

```
// ASSGN SYSxxx,X'cuu',dd,X'ss'
```

SYSxxx is the symbolic unit name and must be one of the following:

SYS000	- Primary output device
SYS001	- Secondary output device
SYS002	- First input device
SYS003	- Second input device
SYS004	- Third input device (used with merge only)
SYS005	- Fourth input device (used with merge only)
SYS006	- Disk drive containing first work area (sort only)
SYS007	- Disk drive containing second work area (sort only)
SYS008	- Disk drive containing housekeeping area (required)

The entry X'cuu',dd specifies the attachment point, c, the unit number, uu, and the device type, dd, for each device. The entries possible are as follows:

X'100',R4	for the 2501 Card Reader
X'200',R5	for the 2520 Card Read-Punch
X'200',R6	for the 2560 MFCM Primary Feed
X'200',R7	for the 2560 MFCM Secondary Feed
X'200',P3	for the 2520 Card Punch
X'300',P2	for the 1442 Card Punch
X'400',L1	for the 1403 Printer
X'400',L3	for the 2203 Printer
X'7uu',T1	for the 2415 7-track Tape Drive uu=00,01,02,...FF
X'7uu',T2	for the 2415 9-track Tape Drive uu=00,01,02,...FF
X'801',D3	for the 2311 Disk Drive, Model 11, Drive 1
X'802',D3	for the 2311 Disk Drive, Model 11, Drive 2
X'801',D4	for the 2311 Disk Drive, Model 12, Drive 1
X'802',D4	for the 2311 Disk Drive, Model 12, Drive 2

The device characteristic X'ss', is used only with tape units (dd = T1 or T2), and ss is replaced by one of the hexadecimal codes in Figure 6. When dd = T1, ss must be specified. When dd = T2, ss need only be specified when all three of the following conditions are met:

1. Tape Drive is Model 4, 5, or 6.

2. The 9-track compatibility special feature is installed.

3. Tape is to be used as output.

ss Codes	Bpi	Parity	Translate Feature	Convert Feature	Tape Tracks
10	200	odd	off	on	7
20	200	even	off	off	7
28	200	even	on	off	7
30	200	odd	off	off	7
38	200	odd	on	off	7
50	556	odd	off	on	7
60	556	even	off	off	7
68	556	even	on	off	7
70	556	odd	off	off	7
78	556	odd	on	off	7
90	800	odd	off	on	7
A0	800	even	off	off	7
A8	800	even	on	off	7
B0	800	odd	off	off	7
B8	800	odd	on	off	7
C0	1600	odd	-	-	9
C8	800	odd	-	-	9

Figure 6. Tape Characteristics

Any symbolic units which might be used but which are not used for a particular job must be unassigned as follows:

```
// ASSGN SYSxxx,UA
```

The following restrictions must be observed when assigning physical units to symbolic unit names (Refer to Figure 31):

1. SYS000-SYS005 may be assigned to disk, tape, or card units.
   SYS006-SYS008 must be assigned to disk units.

2. For output, SYS000 can be disk, tape, card, or UA (the UA applies only when the FREEOUT option is used; see OUTFIL Statement.)
   If SYS000 is on disk, SYS001 must be card or UA.
   If SYS000 is on tape, SYS001 must be alternate tape unit, card, or UA.
   If SYS000 is on card, SYS001 must be UA.

3. For input to a sort, SYS002 may be assigned to a disk, tape, or card unit.
   If SYS002 is on disk, SYS003 must be card or UA.
   If SYS002 is on tape, SYS003 must be alternate tape unit, card, or UA.
   If SYS002 is on card, SYS003 must be UA.

4. For a merge operation, the file assigned to SYS002 is given top priority, SYS003 second, SYS004 third, and SYS005 fourth priority. From one to four input files can thus be assigned to card, tape, or disk, or to tape and card, or to disk and card; both tape and disk assignments are not permitted. Only one file may be asigned to a card device. Alternate tape assignments are not permitted. SYS003, SYS004, and SYS005 must be unassigned (UA) if they are not to be used.

5. For a sort operation, a disk device with a work area must be defined. SYS006 must have the disk assignment if only one work area is used and the first work area assignment if two are used, in which case SYS007 must have the second disk work area assignment. SYS007 must be unassigned (UA) if it is not to be used.

6. For both sorting and merging, a disk device with a housekeeping area must be assigned to SYS008.

DATE Statement

The DATE statement is used to specify the current date and is required, unless a DATE statement has been previously entered. The format of the DATE statement is:

```
   // DATE yyddd
```

The yy is the last two digits of the year (00-99).
The ddd is the three-digit day of the year (001-366).

For example, to specify August 31, 1967, which is the 243rd day of the year, the statement would be as follows:

```
   // DATE 67243
```

CONFG Statement

The CONFG statement defines the amount of core storage available to the system. It is required unless the information is still in core storage from a previous job. The format is:

```
   // CONFG xxx
```

The three-digit entry xxx can be one of the following two values:

101	for 12,288 bytes of core storage
001	for 16,384 bytes of core storage.

For example, a system with 16,384 bytes of core storage would use the following CONFG card:

```
   // CONFG 001
```

LOG Statement

The LOG statement is optional and causes all Job Control statements to be logged (printed). The LOG statement stays in effect until a NOLOG statement is given. The LOG statement can precede the JOB statement. The format is:

```
   // LOG
```

NOLOG Statement

The NOLOG statement is optional and suppresses the logging of Job Control statements until a LOG statement is given. The NOLOG statement can precede the JOB statement. Error messages are always printed, even if NOLOG is given. The format is:

```
   // NOLOG
```

FILES Statement

The FILES statement can be used to position tapes at other than the load point. The FILES statement must immediately follow the ASSGN statement for the symbolic unit containing the tape to be positioned. The format is:

```
   // FILES SYSxxx,n
```

where SYSxxx is the name of the symbolic unit, and n is the number of tape marks to be skipped (from the load point) when positioning the particular tape.

If the FILES statement is used, the entry OPEN=NORWD should be used on the appropriate INPFIL or OUTFIL statement for that tape file. (See INPFIL Statement and OUTFIL Statement.)

EXEC Statement

The EXEC (Execute) statement must be the last card of the Job Control statement deck and is required. The format is:

```
   // EXEC
```

FILE JOB CONTROL STATEMENTS

VOL Statement

Each tape file which uses IBM standard labels must be identified by a VOL (Volume) statement followed immediately by a TPLAB (Tape Label) statement. Each disk file must be identified by a VOL statement followed immediately by both a DLAB (Disk Label) statement and one or more XTENT (Extent) statements. No VOL statement should be given for card files, for unlabelled tape files, or for tape files with non-standard labels. However, a disk Housekeeping or Work area is considered a file. The formats of the possible VOL statements are:

```
   // VOL  SYS000,SORTO    (output file for
                           sort or merge)
   // VOL  SYS002,SORTI    (input to be
                           sorted)
   // VOL  SYS002,FILEA    (first file to
                           be merged)
   // VOL  SYS003,FILEB    (second file to
                           be merged)
   // VOL  SYS004,FILEC    (third file to
                           be merged)
   // VOL  SYS005,FILED    (fourth file to
                           be merged)
   // VOL  SYS006,WORKA    (first disk work
                           area)
   // VOL  SYS007,WORKB    (second disk
                           work area)
   // VOL  SYS008,HSKPG    (disk house-
                           keeping area)
```

TPLAB Statement

A TPLAB statement must be used for tape files with IBM standard labels to identify the file being used. It must immediately follow the appropriate VOL statement and must contain the portions of the standard tape header label associated with the symbolic unit described on that VOL statement. The format of the TPLAB statement is:

```
// TPLAB 'aa...aa'
```

where aa...aa reproduces the 49 characters in fields 3 to 10 of the file label. These 49 characters are interpreted as follows:

Field		Number of Characters
3	File identification	17
4	File serial number	6
5	Volume sequence number	4
6	File sequence number	4
7	Generation number	4
8	Version number of generation	2
9	Creation date	6
10	Expiration date	6

DLAB Statement

All disk files must have IBM standard labels and must be identified by a DLAB statement immediately followed by an XTENT statement. The format of the DLAB statement requires two cards and is:

```
// DLAB 'aa...aa',          c
        ssss,yyddd,yyddd
```

The 'aa...aa' is a string of 51 characters where the first 44 are file identification, one character is format identifier (EBCDIC 1), and 6 characters represent the file serial number (same as volume serial number of first or only pack of the file).

The c is a continuation punch in column 72 so that data can be continued on the second card where columns 1-15 are blank and data starts in column 16.

The ssss is the volume or pack sequence number.

The yyddd,yyddd represents the creation and expiration dates of the file where yy is the last two digits of the year and ddd is the day of the year.

XTENT Statement

Each DLAB must immediately be followed by one or more XTENT cards to identify the high and low extents or limits of the disk areas to be used. No more than 15 extents are permitted for any one volume of an input or output file for the sort or merge operation. The format of the XTENT card is shown in Figure 7.

The 1 is the extent type and, although it is the only type used by the Disk Sort/Merge Program, it must be given for purposes of compatibility.

The sss is a three-digit number from 000 through 255 which gives the sequence number of this extent within a multi-extent file.

The first cccchhh is a seven-digit number which gives the lowest address of the extent where cccc is the cylinder number (0004-0202 for Model 11 and 0004-0102 for Model 12) and hhh is the head number (000-009).

The second cccchhh is a seven-digit number which gives the highest (upper) address of the extent where cccc is the cylinder number and hhh is the head number. It cannot be lower than the first value of cccchhh. Neither value of cccchhh can address the alternate cylinders 1,2, or 3, or track 1 on cylinder 0.

The 'abcdef' is the six-character alphameric volume serial number or name, punched preceded by and followed by an apostrophe. The volume serial number must be identical to the file serial number in the DLAB statement if only one pack is used. Multi-pack input is identified by different volume serial numbers in the XTENT cards. The first extent card must have the same file serial number as the DLAB card.

SYSxxx is the symbolic name of the disk drive.

Housekeeping Area

A disk housekeeping area, HSKPG (SYS008) must be defined by VOL, DLAB, and XTENT statements for both sorting and merging. In addition to housekeeping functions, the HSKPG area is used for checkpoint records.

```
0        1         2         3         4
1        0         0         0         0

// XTENT 1,sss,cccchhh,cccchhh,'abcdef',SYSxxx
```

Figure 7. XTENT Card Format

If the Disk Sort/Merge Program is disk-resident, then one cylinder should be reserved for the housekeeping area. If it is card-resident, four cylinders should be reserved for the housekeeping area.

The housekeeping area cannot be multi-extent.

Work Area

One or two disk work areas, WORKA (SYS006) and WORKB (SYS007), must be specified for sorting applications. If two areas are used, they can be on the same disk drive or on separate drives. VOL, DLAB, and XTENT statements are required for each work area. No work area can be multi-extent.

Each work area must be large enough to accommodate the entire input file plus four tracks. The user should avoid assigning an unnecessarily large area; this not only wastes disk space but may slow the operation. If two work areas are used on the same disk pack, throughput benefits may accrue if they are adjacent. Optimally, two work areas should be on separate drives.

Output Area

Job-Control statements for the output file statements are given only for those output files which are written onto disk or onto tapes with IBM standard labels. No Job Control statements for the output file are to be written for card files or for unlabelled tape files.

If the output file is to be written onto disk, a VOL statement with SORTO (SYS000) must be given followed by a DLAB statement and one or more XTENT statements. The output file may be positioned to overlap one of the work areas in a sort operation if two work areas are specified; no overlapping is permitted if only one work area is specified. (See Figure 25 and Figure 26.)

If the output file is to be written on tape with IBM standard labels, a VOL statement with SORTO (SYS000) must be given followed by a TPLAB statement. The Disk Sort/Merge Program assumes that the output file will be written on the drive with the symbolic address SYS000. To prevent interruption of the output process when the output is to be written on more than one volume, an alternate output drive can be used. The alternate output drive is that drive to which SYS001 is assigned. If SYS001 is unassigned or assigned to a card file, then all tape output will be written on SYS000.

Input Area

Job Control statements for the input file are given only for those input files which come from disk, or from tape with IBM standard labels. No Job Control statements for the input file are to be written for card files, for unlabelled tape files, or for tape files with non-standard labels.

One input file can be specified for sorting and is identified by SORTI (SYS002). Up to four input files can be specified for merging. These files, in order of merging priority, are FILEA (SYS002), FILEB (SYS003), FILEC (SYS004), and FILED (SYS005). Priority specifications are used for merging records whose control fields compare identically. (See ASSGN Statement.)

If an input file comes from tape with IBM standard labels, a VOL statement must be given followed by a TPLAB statement. For sorting only, the initial volume of input data is read from symbolic unit SYS002; if a multi-volume input file is to be read and SYS003 is assigned to a tape drive, then the second volume will be read from SYS003 and the third from SYS002; otherwise all volumes are read from SYS002.

If an input file is read from a disk drive, a VOL statement must be given followed by a DLAB and one or more XTENT statements. The input file to a sort is not permitted to overlap the first work area, WORKA; however, the input file can overlap WORKB. (See Figure 25 and Figure 26.)

Even though the number of disk drives is limited to two, for certain sort cases it may be desirable to use four disk packs. For this type of operation, the following procedure could be used:

WORKA and HSKPG would be on the same pack which could be mounted on the first drive; if the Disk Sort/Merge Program were disk-resident, this would be the pack on which it is located. The input file would be on a second pack and would be mounted on the second drive. After the input file has been read, the pack with the input file would be removed and replaced with the pack containing WORKB. Before the output file is written, the operator would remove the pack containing WORKB and mount the output pack. Each of the operator actions is directed by an appropriate message.

SORT/MERGE CONTROL STATEMENTS

Sort/Merge Control statements are necessary to define the user's specific sort or merge operation. It is the user's responsibility to provide accurate control information describing the files to be sorted or merged, the control fields upon which the records will be sorted or merged, the options to be used, and the modifications to be made. This information is punched into control statement cards, and the cards are inserted into the card reader, in any order, except that the END statement must be the last Sort/Merge Control statement. During the assignment phase, each control statement is analyzed and validity checked for invalid entries and inconsistent combinations of entries. If any errors are detected, a message is printed indicating the nature of each error. Upon completion of the control statement analysis, a halt occurs if any errors were detected. See Appendix D, Diagnostic Messages.

SORT/MERGE CONTROL STATEMENT FORMAT

Each control statement card can contain a maximum of one control statement, although a single statement may require more than one card. The cards must be punched in the following format:

The Name consists of a blank in column 1.

The Operation Code follows the Name and is a mnemonic entry which identifies the nature of the information in the card. It must begin between columns 2 and 15 and be followed by at least one blank.

Operands and Values follow the operation code in any order without embedded blanks, separated from each other by commas. The last operand is followed by a blank. A value expands an operand and can be a numeric entry, an alphameric entry specifying one of two or more alternatives, or an alphameric label for a user-written routine. Each is separated from the operand by an equal sign. If more than one value is given for a particular operand, these should be separated by commas and the whole set of values enclosed in parentheses. Some operands require no values.

Continuation, when information must be continued on another card, is denoted by a non-blank character in column 72. A card following a card with non-blank column 72 must have columns 1-15 blank, with information starting in 16. If column 72 is non-blank, the last punch before column 72 must be a comma.

User Identification may be given in columns 73-80.

```
|0        1        2        3        4 |
|1        0        0        0        0 |
|--------------------------------------|
| SORT     CKPT,FORMAT=FI,FIELDS=(1,3,A) |
```

Figure 8. Control Statement Example

In Figure 8, the operation code is SORT and is in columns 2-5. The three operands are CKPT, FORMAT, and FIELDS. There is no value associated with CKPT, which specifies that the checkpoint feature is to be used. The value FI specifies that fixed-point integer format is used. The values 1,3,A describe the boundaries and sequencing of the comparison field, and, since there is more than one value the set of values is surrounded by parentheses.

Entries are governed by the following additional stipulations:

1. Commas, equal signs, right parentheses, and left parentheses must be used only as delimiters. They cannot be used as values in operands.

2. The termination of an operand is a comma or, in the case of the last operand or value in a statement, a blank. The terminating blank can be in column 72 if the list extends to column 71.

3. Blanks may precede the operation code as long as the first character of the operation code occurs no later than column 15.

4. If a series of equal values is enclosed in parentheses, all of the values after the first can be omitted (the parentheses can also be omitted if only one value is used) and the sort program will assume that they are equal to the last listed value. This does not apply to the LABEL operand where both values must be present. The comma cannot be omitted, unless the value is the last value and a right parenthesis follows; or only one value is left, in which case the parentheses can be entirely omitted. For example, a RECORD statement might contain a single LENGTH value if the input and output records have the same length. The following are identical:

```
LENGTH=(80,80,80)
LENGTH=(80,,80)
LENGTH=(80,80)
LENGTH=(80)
LENGTH=80
```

5. Operands can appear in any order following the operation code.

Sort/Merge Control Statements

The following list indicates the operation codes used:

SORT MERGE	Either SORT or MERGE statement is required
RECORD	Required
INPFIL	Required
OUTFIL	Required
OPTION	Optional
MODS	Optional
END	Required

SORT Statement

The SORT statement is required when sorting is to be done. Either a SORT or MERGE statement must be given. The SORT statement supplies the information as to which data record control fields are to be compared, the order into which the records should be put (called sequencing), and the format in which the data is written. An optional entry gives the possibility of writing checkpoint records.

There are two operands required for every SORT statement (FIELDS and FORMAT) and a third which is optional (CKPT).

The format of the SORT card is shown in Figure 9.

FIELDS uses values of p, m, and s with the following meanings to give the physical boundaries of the control fields and their desired ordering or sequencing:

The p represents a number which gives the location, in bytes, of the left-most position of the control field within the record. The first byte in a record is byte 1, the second is byte 2, and so on.

The m (Measurement) represents a number between 1 and 256 which gives the length of the control field in bytes.

The s represents a letter which denotes the direction of sequencing to be used in sorting the records on that control field. This letter is either A for ascending or D for descending order.

The triplet p1,m1,s1 describes the first or major control field. All records will be compared on the control field of measure or length m1, starting with position p1, and the records will be put into s1 order. If only one field is to be used for comparison within a record, then the form for the operand and its values is:

```
FIELDS=(p1,m1,s1).
```

The triplet p2,m2,s2 describes the second (or first minor) control field, when more than one control field is given. Data contained in the second control field of the two records is compared only when data in the first control fields was identical in the two records. The s2 may be different than s1. Thus, for each field the direction of sequencing is specified independently. In the case where only two fields are given, the form becomes:

```
FIELDS=(p1,m1,s1,p2,m2,s2).
```

The triplet p3,m3,s3 describes the third control field. This would be used only when the first two control fields in one record were equal to the first two control fields in another record. In the case where only three fields are given the form becomes:

```
FIELDS=(p1,m1,s1,p2,m2,s2,p3,m3,s3).
```

From 1 to 12 control fields can be specified. The triplet p12,m12,s12 describes the twelfth control field and would be used only if the other eleven fields in a pair of records were identical. In this case the form becomes:

```
FIELDS=(p1,m1,s1,p2,m2,s2,p3,m3,s3,p4,m4,
        s4,p5,m5,s5,p6,m6,s6,p7,m7,s7,p8,
        m8,s8,p9,m9,s9,p10,m10,s10,p11,m11,
        s11,p12,m12,s12)
```

```
|0          1          2          3          4          5          6    |
|1          0          0          0          0          0          0    |
|-----------------------------------------------------------------------|
| SORT FIELDS=(p1,m1,s1,p2,m2,s2,...,p12,m12,s12),FORMAT=xx,CKPT       |
```

Figure 9. SORT Card Format

This can be written in a generalized form to indicate from 1 to 12 control fields:

FIELDS=(p1,m1,s1,p2,m2,s2,...,p12,m12,s12)

Regardless of how many control fields are given, the total measurement or length of all control fields combined cannot be greater than 256 bytes. Control fields are not allowed to overlap except for those with binary or alphameric format.

FORMAT, with one of five values, gives the format of the data contained in all control fields. The value is given by one of the two-character symbols BI, CH, FI, PD, and ZD.

BI refers to binary format; that is, the fields are then considered to contain unsigned binary data.

CH refers to character format; that is, the fields are considered to contain alphameric data.

FI refers to fixed-point integer format. That means that fields are in signed binary format; i.e., they are assumed to have the sign given in the left-most bit of the left-most byte of the field.

PD refers to packed-decimal format. This assumes that each data byte contains two four-bit decimal digits (0-9), with the exception of the right half of the right-most byte which contains the code for the sign.

ZD refers to zoned-decimal format; that is, each data byte is assumed to contain one four-bit decimal digit (0-9) in the right half of each byte. The left half (zone portion) of each byte is assumed to contain a four-bit code. The zone code in the right-most byte gives the sign of the data.

If a control field containing data in FI, PD, or ZD format has the same sign (all positive or all negative) in all records, and the sign consists of the same bit structure in all records, then that control field may be considered binary (BI) or alphameric-character (CH) format. Specifying a FORMAT of CH or BI saves processing time and core storage as no conversion routines are needed. Care must then, however, be taken to specify -- for a negative PD or ZD control field -- the reverse sequence from that which is desired (i.e., descending sequence if ascending algebraic sequence is desired, and vice versa).

All data in all control fields described under FIELDS is considered to have the same format. If the user wishes to sort with more than one format, more than one sorting run can be used. The lowest-ranking control field must first be sorted using the appropriate format, followed by the next-lowest control field and its format, and so on.

CKPT has no value associated with it. If CKPT is given, then intermediate checkpoint records are written to permit restarting, should it become necessary to interrupt during execution. If CKPT is not given, the intermediate checkpoint records are not written.

The operands with their corresponding values for SORT statements are then:

```
FIELDS=(p1,m1,s1,p2,m2,s2,...,p12,m12,s12)
FORMAT=BI or CH or FI or PD or ZD
CKPT
```

In the examples shown in Figure 10, the major control field extends from bytes 2 to 6 and records are to be placed into ascending order. In all except the last example, the format of the data in the control fields is character or binary; that is, the fields have no arithmetic sign and all bits enter into the comparison.

In examples 2 and 3 of Figure 10 the second control field extends from bytes 46 to 48 and records are to be placed into descending order on that control field. In example 3, the third control field extends from bytes 10 to 50 and records are to be placed into descending order on that control field. The automatic checkpoint option is invoked in this case. Control field 3 overlaps control field 2. This is permissible because the format is CH.

```
    ┌─────────────────────────────────────────────────┐
    │0        1         2         3         4        5│
    │1        0         0         0         0        0│
    ├─────────────────────────────────────────────────┤
1.  │ SORT FIELDS=(2,5,A),FORMAT=CH                   │
2.  │ SORT FIELDS=(2,5,A,46,3,D),FORMAT=CH            │
3.  │ SORT FIELDS=(2,5,A,46,3,D,10,41,D),FORMAT=CH,CKPT│
4.  │ SORT FIELDS=(2,5,A,10,251,D),FORMAT=FI          │
    └─────────────────────────────────────────────────┘
```

Figure 10. SORT Statement Examples

```
┌─────────────────────────────────────────────────────────┐
│0        1         2         3         4         5       │
│1        0         0         0         0         0       │
├─────────────────────────────────────────────────────────┤
│ MERGE FIELDS=(p1,m1,s1,...,p12,m12,s12),FORMAT=xx,ORDER=n│
└─────────────────────────────────────────────────────────┘
```

Figure 11. MERGE Statement Format

In the fourth example in Figure 10, the major control field extends from bytes 2 to 6, and the minor control field extends from bytes 10 to 260. The total measure or length of all control fields is 256, which is the maximum allowable total length. The format of the data in these fields is assumed to be fixed-point integer, that is, the sign is to be considered in sorting and the signs are given in the left-most bits of bytes 2 and 10.

MERGE Statement

The MERGE control statement is required when merging is to be done. Either a SORT or a MERGE statement must be present in any application. The form is similar and in part identical to that of the SORT statement.

There are three operands for a MERGE statement, all of which are required: FIELDS, FORMAT, and ORDER. The first two are as described under SORT Statement. No restart procedure is available with the merge program and therefore checkpoint records are not written. The format of the MERGE statement is shown in Figure 11.

ORDER has a value, n, which is a number from 1 to 4, giving the number of files to be merged. If n is 2, 3, or 4, the files are compared on the control fields and sequenced into one large file which is in the same order as the input files. The files to be merged are assumed to be already in order. If n is 1, then the file is copied, checked to be sure that it is in the correct sequence, and the option is given for reblocking the data while copying the file. If the sequencing is incorrect, an error message is printed followed by a stop.

When merging, if two records of different input files compare identically in all control fields, the record with the higher priority is listed first on the output file. Priority is based on the ASSGN statements. Precedence is in the following order: FILEA (SYS002), FILEB (SYS003), FILEC (SYS004), and FILED (SYS005).

The operands with their corresponding values for MERGE statements are then:

```
FIELDS=(p1,m1,s1,p2,m2,s2,...,p12,m12,s12)
FORMAT=BI or CH or FI or PD or ZD
ORDER=1 or 2 or 3 or 4
```

In the example shown in Figure 12 the designations of the control fields and formats are identical with those described under SORT Statement for the examples in Figure 10, except that they control the merging of files instead of the sorting within a file. In the first example, 4 files are to be merged. In the second example, 3 files are to be merged. In the third, 2 files are to be merged. In the last, one file is to be copied and simultaneously checked to be sure that the records in the file are in the correct order.

RECORD Statement

The RECORD statement is required. It gives the length of records to be used and allows the option of removing records which contain a specified character in a specified position within the record. There are two required operands, TYPE and LENGTH, and one optional operand, DELBLANK.

The format of the RECORD statement is in Figure 13.

```
    ┌─────────────────────────────────────────────────────────┐
    │0        1         2         3         4         5       │
    │1        0         0         0         0         0       │
    ├─────────────────────────────────────────────────────────┤
1.  │ MERGE FIELDS=(2,5,A),FORMAT=CH,ORDER=4                  │
2.  │ MERGE FIELDS=(2,5,A,46,3,D),FORMAT=CH,ORDER=3           │
3.  │ MERGE FIELDS=(2,5,A,46,3,D,10,41,D),FORMAT=CH,ORDER=2   │
4.  │ MERGE FIELDS=(2,5,A,10,251,D),FORMAT=FI,ORDER=1         │
    └─────────────────────────────────────────────────────────┘
```

Figure 12. MERGE Statement Examples

```
┌──────────────────────────────────────────────┐
|0        1         2         3         4     |
|1        0         0         0         0     |
├──────────────────────────────────────────────┤
| RECORD TYPE=F,LENGTH=(ℓ1,ℓ2,ℓ3),DELBLANK=(p,q) |
└──────────────────────────────────────────────┘
```

Figure 13. RECORD Statement Format

TYPE=F gives the record TYPE of Fixed length. This is the only type of record processed; however, for purposes of compatibility it must be given.

The letter ℓ is used in Figure 13 to signify that a length value is to be substituted. In the following text the word length is used rather than ℓ.

LENGTH gives the lengths length1, length2, length3 (explained below) and uses the form:

LENGTH=(length1,length2,length3)

The length1 is the length in bytes of the input record and is required.

The length2 is the length in bytes of each intermediate record generated during sorting and can be no larger than length1. If length2 is identical to length1, the length2 can be omitted by using the form:

LENGTH=(length1,,length3)

The length2 is ignored in a merge-only operation.

The length3 is the length in bytes of the output record and can be no larger than length2 for a sort application and no larger than length1 for a merge-only application. If length3 is identical to length2, the following form can be used:

LENGTH=(length1,length2)

If all three lengths are identical, the following form can be used:

LENGTH=length1

The three lengths must retain the relationship:

length3 ≤ length2 ≤ length1

Output records can be shortened through a user-written subroutine (see MODS Statement), or by simply giving length3 less than length2 which causes output records to be truncated or cut off at the right-hand end of the data to shorten the records to the length length3.

In a merge-only operation length2 can be entirely omitted and the following could be used if length3 is different from length1:

LENGTH=(length1,,length3)

If length3 is great enough to contain all control fields, the following format may also be used: LENGTH=(length1,length3)

DELBLANK, when used, causes records to be deleted which contain a certain character in a specified byte position. That position is called the blank record indicator since the usual character is a blank. The option is usually used to delete blank or separator records in a disk or tape file. The p is a number giving the position of the byte within each record which serves as the blank record indicator. The comparison character replaces q. If "blank" is the comparison character then q is eliminated. The form of the statement is one of the following:

DELBLANK=(p,q)
DELBLANK=p

The operands with their corresponding values for RECORD statements are then:

TYPE=F
LENGTH=(length1,length2,length3)
DELBLANK=(p,q)

The first four examples in Figure 14 are equivalent; all specify input, intermediate, and output lengths of 80 bytes. In the fifth example, the input and intermediate record lengths are 80 bytes, but the output records are truncated automatically, or compressed by a user-written routine, to 75 bytes. In the sixth example, the input record length is 80 bytes; the intermediate record length is 78 bytes, and the output records are 75 bytes long.

The seventh, eighth, and ninth examples in Figure 14 cause certain records to be deleted. The length of all records is 140 bytes. The seventh example causes all records to be deleted which contain a blank in the first byte of the record. The eighth example deletes all records with the letter "Z" in the 35th byte of the record. The ninth example deletes all records with the character "+" in the last byte of the record.

```
      |0       1       2       3       4       5   |
      |1       0       0       0       0       0   |
      |--------------------------------------------|
1.    | RECORD TYPE=F,LENGTH=(80,80,80)            |
2.    | RECORD TYPE=F,LENGTH=(80,80)               |
3.    | RECORD TYPE=F,LENGTH=(80,,80)              |
4.    | RECORD TYPE=F,LENGTH=80                    |
5.    | RECORD TYPE=F,LENGTH=(80,,75)              |
6.    | RECORD TYPE=F,LENGTH=(80,78,75)            |
7.    | RECORD TYPE=F,LENGTH=140,DELBLANK=1        |
8.    | RECORD TYPE=F,LENGTH=140,DELBLANK=(35,Z)   |
9.    | RECORD TYPE=F,LENGTH=140,DELBLANK=(140,+)  |
```

Figure 14. RECORD Statement Examples

```
|0       1       2       3       4       5       6       7       |
|1       0       0       0       0       0       0       0       |
|----------------------------------------------------------------|
| INPFIL BLKSIZE=(n,X),OPEN=n,CLOSE=n,VOLUME=(na,nb,nc,nd),BYPASS,      C |
|              SKIPBYTE=n,PRESEQ                                  |
```

Figure 15. INPFIL Statement Format

All examples in Figure 14 contain the entry TYPE=F to denote fixed-length records.

INPFIL Statement

The INPFIL statement is required and describes the input data file(s) to be processed, as well as the size of the input blocks. There are also several options which may be used.

There is one required operand BLKSIZE, and there are three optional operands which apply only to tape; OPEN, CLOSE and VOLUME; two operands for tape and disk, BYPASS and SKIPBYTE; and one optional operand for any type of input, PRESEQ.

The format of the INPFIL statement is shown in Figure 15.

BLKSIZE uses a value for n which specifies the number of bytes per input block, and a character X which indicates that fixed-length blocks are used. (For tape input, the number of records per block is permitted to vary. The value n must then represent the maximum number of bytes in a block. The character X must always be given.) If a disk file is specified and n is not a multiple of 270, n is automatically increased to the next higher multiple of 270 so that the block size will be a number of complete sectors.

If card processing alone is specified, n must be no larger than 80. If card processing is specified in combination with either disk or tape, n refers to the block size of the disk or tape file; n is automatically reduced to 80 for processing the card file except when n was already 80 or less for the disk or tape file, in which case n is unaltered for processing the card file.

It is not necessary that n be a multiple of the input record length given in the RECORD statement. For instance, if the block size were 540 and the record length were 70, then there would be 7 records per block with 50 unused bytes in each block. If a card file were to be processed as well as disk file with block size of 540 and record length of 70, then only the first 70 columns of each card would be processed.

The form of the block-size operand and its corresponding values is:

BLKSIZE=(n,X)

OPEN provides the choice of rewinding or not rewinding the first (or only) tape in a tape file before reading it. The two possible values are RWD (rewind without unloading) and NORWD (no rewind). If NORWD is specified, it applies only to the first tape in the file; all others will be rewound before being read. RWD is assumed if this entry is omitted. For a merge-only operation, the OPEN value applies to the first (or only) volume of each file. The entry must be one of the following:

OPEN=RWD
OPEN=NORWD

All input tapes except the first are always rewound before being read, and all input tapes except the last are rewound and unloaded after being read. If the user specifies NORWD, he is responsible for seeing that the tape is in the correct position.

CLOSE provides the choice of rewinding with or without unloading or of not rewinding the last (or only) tape of the file after it has been processed. The three possible values are RWD (rewind without unloading), UNLD (rewind and unload), and NORWD (no rewind). If NORWD or RWD is specified, it applies only to the last tape in the file; all others will be rewound and unloaded. For a merge-only operation, the CLOSE value applies to the last (or only) volume of each file. RWD is assumed if this entry is omitted. The entry must be one of the following:

```
CLOSE=RWD
CLOSE=UNLD
CLOSE=NORWD
```

VOLUME specifies the number, n, of volumes (reels) of tape to be processed for each input file. For merge-only, there can be up to four input files, and each file can have a number of volumes given by a number na for FILEA, nb for FILEB, nc for FILEC, and nd for FILED. The form of the entry for merging four files is:

```
VOLUME=(na,nb,nc,nd)
```

If all files have the same number of volumes the following shorter form can be used, or, if sorting is to be done, the following form must be used:

```
VOLUME=n
```

Any of the values nb, nc, nd could be omitted if it is the same number as the number preceding it; the comma cannot be omitted unless the value to be omitted is the last value in the set. That is, if four files were to be merged and the first three files had 1 volume each and the fourth file had 4 volumes, the entry would be one of the following:

```
VOLUME=(1,,,4)
VOLUME=(1,1,1,4)
VOLUME=(1,1,,4)
VOLUME=(1,,1,4)
```

If the first file had 4 volumes and the second, third, and fourth files had one volume each, then any of the following entries could be used:

```
VOLUME=(4,1,1,1)
VOLUME=(4,1,1)
VOLUME=(4,1,,1)
VOLUME=(4,1)
```

If the VOLUME entry is omitted and nonstandard labels are specified, one volume per file will be processed. If the VOLUME entry is omitted and IBM standard labels are specified, input records will be processed until an end-of-file trailer label is read. If the VOLUME entry is given and standard labels are processed, input records will be processed until either an end-of-file trailer label is read or the specified number of volumes has been read, whichever comes first. For a merge-only operation this is done for each file.

BYPASS is a an operand with no associated values which applies to disk and tape input and allows permanently unreadable blocks to be bypassed on input. That is, a data block with incorrect format or invalid characters would be bypassed on input without being processed. If BYPASS is omitted and an unreadable block occurs, an error message is printed, followed by a halt. If BYPASS is used, the number of unreadable blocks which were bypassed will be printed. (See Appendix C, Exit 13.)

SKIPBYTE is used when input records from disk or tape do not start in the first byte of a block but rather in the (n+1) st byte. This option applies only to sorting applications. When, for example, input comes from a disk drive and the block size or the sum total of all bytes in the records is not a multiple of 270 bytes, there are extra bytes in each block not used by the data. If some or all of these bytes occur at the start of a block, then the SKIPBYTE option would be used where n gives the number of bytes to be skipped at the start of each input block. The input block size must include the bytes to be skipped by the SKIPBYTE option. If this option is omitted, the records are assumed to start with the first byte of each block. The form of the entry is:

```
SKIPBYTE=n
```

PRESEQ should be used where there is appreciable pre-sequencing of records present in the input file. It may make the sorting operation considerably faster.

Operands and their corresponding values for the INPFIL statement are:

```
BLKSIZE=(n,X)
OPEN=RWD or NORWD
CLOSE=RWD or UNLD or NORWD
VOLUME=n or (na,nb,nc,nd)
BYPASS
SKIPBYTE=n
PRESEQ
```

```
     0         1         2         3         4         5
     1         0         0         0         0         0
1.   INPFIL BLKSIZE=(80,X)
2.   INPFIL BLKSIZE=(50,X)
3.   INPFIL BLKSIZE=(270,X)
4.   INPFIL BLKSIZE=(540,X)
5.   INPFIL BLKSIZE=(2700,X),SKIPBYTE=20
```

Figure 16. INPFIL Statement Examples

```
     0         1         2         3         4         5
     1         0         0         0         0         0
1.   INPFIL BLKSIZE=(400,X),OPEN=RWD,CLOSE=RWD
2.   INPFIL BLKSIZE=(400,X),OPEN=NORWD,CLOSE=UNLD
3.   INPFIL BLKSIZE=(800,X),BYPASS
4.   INPFIL BLKSIZE=(800,X),VOLUME=3,CLOSE=NORWD
5.   INPFIL BLKSIZE=(800,X),VOLUME=(1,2,3,4)
6.   INPFIL BLKSIZE=(800,X),VOLUME=(1,2,3)
7.   INPFIL BLKSIZE=(800,X),VOLUME=(1,,,4)
```

Figure 17. INPFIL Statement Examples for Tape

All examples in Figure 16 could apply equally well to disk or tape, and the first two examples could apply to card input.

In the first example in Figure 16, input is from cards with all 80 columns read or from disk or tape. In the second example, input is from card with the first 50 columns only being read or from disk or tape. In the third example, data is read in blocks of one sector from the disk (270 bytes), or from tape blocks which are a maximum of 270 bytes. In the fourth example, data is read in blocks of two sectors from the disk (540 bytes), or from tape blocks which are a maximum of 540 bytes. In the fifth example, data is read in blocks of 10 sectors from the disk (2700 bytes), or from tape blocks with a maximum of 2700 bytes, the first record of each block starts in byte 21 instead of byte 1. An unreadable input block will cause a halt unless BYPASS or Exit 13 has been specified.

All examples in Figure 17 apply to tape, and the third example applies also to disk. In all of the examples, all except the first input tape of each file will be rewound before being read, and all except the last input tape of each file will be rewound and unloaded after being read. In the second example, the first input tape is not rewound before being read, but the last input tape is rewound and unloaded after being read. The first two examples process tape blocks of 400 bytes.

The third example in Figure 17 processes files with 800 bytes per block. Unreadable records are bypassed without being processed. Unreadable records encountered when processing the other examples in Figure 17 cause an error message to be printed followed by a halt. The default value of RWD is assumed for both OPEN and CLOSE.

In the fourth example, the first input tape is rewound before being read but the last input tape is not rewound after being read. The fourth example in Figure 17 processes three volumes for sorting or three volumes per file for merging; the first two volumes are rewound after being read, but the third volume is not.

The fifth, sixth, and seventh examples apply only to merging. In the fifth example, one volume is processed for FILEA, two volumes for FILEB, three volumes for FILEC, and four volumes for FILED. The sixth example is similar to the fifth example except the FILED either has the same number of volumes as FILEC, or is not given, or comes from cards (the block size is automatically reduced to 80 to process the card file). In the seventh example, one volume each is processed for the first three files and four volumes for FILED. In the fourth through the seventh example, if IBM standard labels are specified, each file is terminated when either the specified number of volumes has been read or an end-of-file trailer label has been read, whichever comes first.

OUTFIL Statement

The OUTFIL statement is required and describes the output file block size. Some options are also available.

There is one required operand, BLKSIZE, two optional operands for tape only, OPEN and CLOSE, and one optional operand for disk sort output only, FREEOUT.

The format of the OUTFIL statement is one of the following:

```
OUTFIL BLKSIZE=n,OPEN=n,CLOSE=n
OUTFIL BLKSIZE=n,FREEOUT
```

BLKSIZE uses a number, denoted by n, which indicates the number of bytes per output block. The use of BLKSIZE is as explained under INPFIL Statement, except that the value X is not used. The format is:

```
BLKSIZE=n
```

OPEN and CLOSE are tape options which function as described under INPFIL Statement, except that they control action taken on the output tape before and after writing instead of on the input tape before and after reading. The possible entries are:

```
OPEN=RWD
OPEN=NORWD
CLOSE=RWD
CLOSE=UNLD
CLOSE=NORWD
```

FREEOUT is an optional operand, without any corresponding values, used only when output is to disk and two work areas are specified. When specified, the Disk Sort/Merge Program is given the option of selecting and utilizing WORKA or WORKB as the final output file. This could save an extra pass when the output area is to overlap one of the work areas. FREEOUT cannot be used for merging since no work areas are specified. If FREEOUT is used, SYS000 must be UA (unassigned) and no VOL, DLAB or XTENT statement may be specified. The location of the output file (in the SYS006 or SYS007 extent) will be identified by a message at the beginning of phase 3.

Operands and their corresponding values for the OUTFIL statement are:

```
BLKSIZE=n
OPEN=RWD or NORWD
CLOSE=RWD or UNLD or NORWD
FREEOUT
```

```
    0         1         2         3
    1         0         0         0
1.   OUTFIL BLKSIZE=80
2.   OUTFIL BLKSIZE=50
3.   OUTFIL BLKSIZE=270
4.   OUTFIL BLKSIZE=540
5.   OUTFIL BLKSIZE=2700,FREEOUT
```

Figure 18. OUTFIL Statement Examples

The five examples in Figure 18 correspond to the five examples in Figure 16 for the input file, where those in Figure 18 are for the output file. Blank columns (for card output to the end of the card) and blank characters (for disk or tape output) are added to the output data at the end of the block. In addition, the fifth example, which applies only to disk sort, allows the Disk Sort/Merge Program to choose the optimum location within the work area for the output file when the output is on disk and is to overlap one of the work areas after sorting.

The six examples in Figure 19 apply to tape, the first and fourth examples could, however, also apply to disk.

The first two examples in Figure 19 are equivalent; tapes are rewound before being written, all except the last output tape are rewound and unloaded after being written, and the last output tape is rewound but not unloaded after being written. The third example rewinds all except the first output tape before writing but does not rewind the first before writing; all tapes are rewound and unloaded after being written.

The fourth and fifth examples are like the first example except that the block size is 800 instead of 400. The sixth example in Figure 19 rewinds all tapes before writing and rewinds and unloads all tapes except the last (or only) output tape after writing; the last tape is not rewound after being written.

MODS Statement

The MODS statement is optional and indicates that user-written routines have been added to the Sort/Merge Program. This control statement supplies the main program with the symbolic name and length of each user-written routine. Also specified is the exit or exits used to branch out of the main-line program into the user's routine.

```
    ┌─────────────────────────────────────────────────────────────────────┐
    |0         1           2           3           4           5          |
    |1         0           0           0           0           0          |
    ├─────────────────────────────────────────────────────────────────────┤
1.  | OUTFIL BLKSIZE=400                                                  |
2.  | OUTFIL BLKSIZE=400,OPEN=RWD,CLOSE=RWD                               |
3.  | OUTFIL BLKSIZE=400,OPEN=NORWD,CLOSE=UNLD                            |
4.  | OUTFIL BLKSIZE=800                                                  |
5.  | OUTFIL BLKSIZE=800,OPEN=RWD                                         |
6.  | OUTFIL BLKSIZE=800,CLOSE=NORWD                                      |
    └─────────────────────────────────────────────────────────────────────┘
```

Figure 19. OUTFIL Statement Examples for Tape

User-written routines (UWSn) can be used only in specific cases in phases 1 and 3 for sorting and phase 4 for merging. See User-Written Routines and Program Description for a complete description of the procedures of each phase and the exact use of each exit.

There are three operands possible: PH1 (for PHase 1), PH3 (for PHase 3), and PH4 (for PHase 4).

There are three values for each operand: name, length, and exit. The format of the MODS statement is:

```
┌──────────────────────────────────────────────────────┐
|0          1           2           3            4   |
|1          0           0           0            0   |
├──────────────────────────────────────────────────────┤
| MODS PHx=(name,length,exit,...,exit)                 |
└──────────────────────────────────────────────────────┘
```

The name is the symbolic name of the user-written routine used for that phase. The name must be one to six characters long; the first character must be a letter, and embedded blanks and special characters are not permitted. The name must be omitted (but the comma which follows may not be omitted) if the user-written routine is to be loaded from cards. The name must be given if the user-written routine is to be loaded from disk.

The length is the total number of bytes of core storage to be occupied by the modification program (user-written routine) for the particular phase. This must include the associated branch table plus any necessary half-word adjustment.

The exit identifies the Sort/Merge Program exit from which name is to be accessed. Each exit is identified by an E followed by two digits, where the first digit gives the program phase in which the exit occurs and the second digit identifies the exit within the phase. For phase 1, exit can be E11 and/or E12 and/or E13. For phase 3, exit can be E31 and/or E32. For phase 4, exit can be E41 and/or E42 and/or E43 and/or E44.

The maximum possible entries in the MODS statement are:

PH1=(name,length,E11,E12,E13) and
PH3=(name,length,E31,E32) or
PH4=(name,length,E41,E42,E43,E44)

The first three examples in Figure 20 apply to sorting applications. The first example specifies that UWS1, which occupies 554 bytes of core storage, is accessed during phase 1 through exit 11 only. The second example uses UWS1 but accesses it through exit 12 only. UWS3, which occupies 442 bytes of core storage, is accessed during phase 3 through exit 31 only. The third example uses UWS1 and UWS3 and utilizes four exits; during phase 1,554 bytes are required and during phase 3,442 bytes are required.

```
    ┌─────────────────────────────────────────────────────────────────────┐
    |0         1           2           3           4           5          |
    |1         0           0           0           0           0          |
    ├─────────────────────────────────────────────────────────────────────┤
1.  | MODS PH1=(UWS1,554,E11)                                             |
2.  | MODS PH1=(UWS1,554,E12),PH3=(UWS3,442,E31)                          |
3.  | MODS PH1=(UWS1,554,E11,E12),PH3=(UWS3,442,E31,E32)                  |
4.  | MODS PH4=(UWS4,300,E41)                                             |
5.  | MODS PH4=(UWS4,300,E41,E42,E43)                                     |
    └─────────────────────────────────────────────────────────────────────┘
```

Figure 20. MODS Statement Examples

```
┌──────────────────────────────────────────────────────────────────────────────┐
│0       1       2       3       4       5       6       7                     │
│1       0       0       0       0       0       0       0                     │
├──────────────────────────────────────────────────────────────────────────────┤
│ OPTION PRINT,STORAGE=n,LABEL=(i,o),ADDROUT,VERIFY,NOTPMK,RESTART,     C      │
│                TIME=(n1,n2)                                                  │
└──────────────────────────────────────────────────────────────────────────────┘
```

Figure 21. Format of OPTION Statement

The fourth and fifth examples in Figure 20 apply to merging applications. The fourth example specifies that UWS4, which occupies 300 bytes of core storage, is accessed during phase 4 through exit 41 only. The fifth example uses UWS4 and utilizes three exits.

OPTION Statement

The OPTION statement is optional and is used to specify from one to eight options which are to be used. The options can be given in any order.

The eight options with their associated values, if any, are the following:

PRINT
STORAGE=n
LABEL=(i,o)
ADDROUT
VERIFY
NOTPMK
RESTART
TIME=(n1,n2)

The format of the OPTION statement is shown in Figure 21 (all entries cannot be made simultaneously):

PRINT specifies that pass-count messages and record counts are to be printed. Even if this option is omitted, diagnostic messages, phase-end and Sort/Merge control card messages are printed.

STORAGE=n specifies the number of positions of core storage available for the Disk Sort/Merge Program and user modifications, if any. The n may be any number between 12286 and 16384. If this statement is omitted, all core storage in the system as specified in the CONFG statement is assumed to be available. This operand is used only if the upper part of core storage contains information that is not to be disturbed by the Disk Sort/Merge Program.

LABEL=(i,o) is required if tape files are to be processed with other than standard labels; i refers to input labels and is to be replaced by S, U, or N; o refers to output labels and is to be replaced by S or U.

S = Standard labels
U = Unlabelled Files
N = Non-standard input tape labels (permits non-standard tape input labels to be bypassed but not processed)

If this operand is used, both values must be specified. If either input or output does not include a tape file, the value S must be used as the label type specification for that file.

If this operand is omitted, standard labels will be assumed for both input and output tape files. All disk files are required to have standard labels, and card files are unlabelled.

ADDROUT specifies that output from a sort is to be a list of disk addresses of the sorted records rather than the sorted records themselves. This option can be used only with single-pack disk input (not in combination with card input). The output file consists of one contiguous string of blocked addresses in the output area. The format in hexadecimal notation of each output address is:

mcchhrbb where
 m = pack number = 00
cc = cylinder number
hh = read/write head number
 r = sector number (or first sector for block which occupies several sectors)
bb = zero for unblocked records; or the displacement, in bytes, of the record within the block.

It should be noted that, regardless of what the user specifies, length2 and length3 to be in the RECORD statement, length3 will be equal to 8 bytes, and length2 will be equal to 8 bytes plus the total number of bytes in all control fields. In the typical case, the use of ADDROUT will result in much smaller disk work area and output area requirements as well as faster performance of the sort.

VERIFY specifies that the disk output file will be program verified; that is, the data written on the disk will be compared with the data in core to ensure that no discrepancies exist. If this option is omitted, no verification will be performed. Although the use of the VERIFY option will result in somewhat slower operation, it is highly recommended.

NOTPMK specifies that no tape mark should be written before the first data record in a tape output file. A tape mark is normally automatically written before the first data record in a tape output file. NOTPMK should be used only when an unlabelled tape output file is to be written. If NOTPMK and tape with IBM standard labels are specified, a diagnostic message will be printed followed by a halt.

RESTART is specified only if a sort run has been previously interrupted and this is to be a restart run from the last checkpoint. This option applies only to sort applications when two work areas have been specified. All control statement cards from the original run must be re-entered with the addition of the RESTART operand in the OPTION card. (See Checkpoint and Restart.)

TIME=(n1,n2) is specified if the desired output is a set of timing estimations rather than the actual sorted output. This option is used for timing sort operations (phase 5). The n1 specifies the minimum file size (number of records) for which an estimate is desired; n2 specifies the desired increment between file sizes. Timing estimates will then be provided for ten file sizes, which are n1,n1+n2,n1+2n2,...,n1+9n2. All areas (housekeeping, work, input, output) must be specified, but only the pack with the housekeeping area need be mounted.

```
    ┌──────────────────────────────────────┐
    │0         1         2         3       │
    │1         0         0         0       │
    ├──────────────────────────────────────┤
1.  │ OPTION PRINT                         │
2.  │ OPTION PRINT,STORAGE=16300           │
3.  │ OPTION LABEL=(N,U),NOTPMK            │
4.  │ OPTION LABEL=(U,S)                   │
5.  │ OPTION ADDROUT,PRINT                 │
6.  │ OPTION VERIFY,RESTART                │
7.  │ OPTION TIME=(1000,200),PRINT         │
    └──────────────────────────────────────┘
```

Figure 22. OPTION Statement Examples

The examples in Figure 22 show various combinations of options. The first, second, fifth, and seventh example cause end of pass and record count messages to be printed in addition to diagnostic messages and phase-end messages.

The second example in Figure 22 applies only to a 16K machine configuration, where core positions 16300 through 16383 contain data not to be disturbed.

The third example in Figure 22 applies to tape files; it assumes that input tapes have non-standard labels and that output tapes are to be unlabelled with no tape mark at the start of the tape. The fourth example assumes that input is either from unlabelled tape files or card files, but that standard labels are to be written on the output tape.

The fifth example in Figure 22 applies only to disk input; addresses of the sorted records rather than the records themselves are written on the output file.

The sixth example in Figure 22 applies only to disk output and specifies that output data is to be verified. In addition, the sixth example is for a sort application which was previously interrupted and is to be restarted.

The seventh example is not used for actual sorting but to obtain timing estimates for sorting files of sizes 1000, 1200, 1400,...,2800 records using the specifications given in the Job Control and Sort/Merge Control statements.

END Statement

The END statement must be the physically last of the control statements for a particular application. There are no associated values. The form is given in Figure 23.

If batch processing is desired, a blank card, an EOF card, or a last card condition must follow.

```
┌─────────────────────────────────────────┐
│0         1         2         3        4 │
│1         0         0         0        0 │
├─────────────────────────────────────────┤
│ END                                     │
└─────────────────────────────────────────┘
```

Figure 23. END Statement Example

The Disk Sort/Merge Program is divided into phases as follows:

Assignment phase	(Phase 0)
Internal Sort phase	
for partially presequenced input	(Phase 1A)
for random input	(Phase 1B)
External Sort phase	
Variable string interleave	(Phase 2A)
Uni-resident sort	(Phase 2B)
Final Sort phase	
Variable string interleave	(Phase 3A)
Uni-resident sort	(Phase 3B)
Merge-Only phase	(Phase 4)
Timing phase	(Phase 5)

If sorting is to be performed, phases 0, 1 (A or B), 2 (A or B), and 3 (A or B) are executed. User-written modifications may be added to phases 1 (A or B) and 3 (A or B).

If merging is to be performed, phases 0 and 4 are executed. User-written modification routines may be added to phase 4.

If a timing estimate for sorting is to be performed, phases 0 and 5 are executed. Program flow is illustrated in Figure 24.

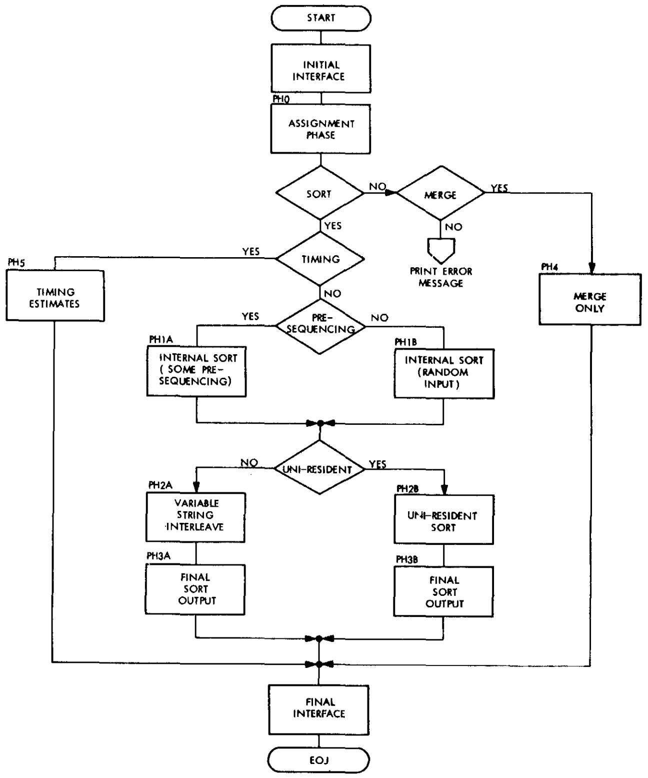


Figure 24. Flowchart of the Disk Sort/Merge Program Phases

PHASE 0 - ASSIGNMENT PHASE

The assignment phase processes the Sort/Merge Control statements and performs the necessary computations, makes the decisions, and compiles the data for the initialization of each phase. It performs a diagnostic check on the control information, checking for such things as missing control statements and duplicate or invalid operands.

In the event an error is detected in the control statements, it is identified by an appropriate message (See Appendix D, Diagnostic Messages). Then, after all statements are edited, a halt occurs and the user must correct the errors and reload.

IBM standard labels are processed for the disk housekeeping area, disk input file(s) (if any), and the first work area if sorting is to be done.

PHASE 1 - INTERNAL SORT

During Phase 1 the records are read into the core storage input area and are sorted into strings that are at least G (the maximum number of records that can be sorted internally at one time) in length. These strings, which are variable in length, are written out on the 1316 disk pack work area.

Phase 1 allows multi-volume tape or disk input files. It also allows input from cards, alone or in combination with either disk or tape. IBM standard labels are processed for the input file if tape files with IBM standard labels or multi-pack disk input files are specified.

Phase 1 provides exits 11, 12, and 13 which may be activated to access user-written routines. These routines may be employed to make revisions in the format of input blocks or input records prior to their being processed by the sort, or to process "user" input tape labels, or to translate data, or to process unreadable blocks.

PHASE 2 - EXTERNAL SORT

If two work areas were specified, IBM standard labels are checked for the second work area.

The first pass of the external sort phase retrieves the ordered strings, which were created in phase 1, merges them, and writes the longer strings out to the disk file. The output of each pass becomes the input to the following pass which creates still longer strings. This process continues until the number of strings is reduced to the order of merge.

Although the same internal technique of merging is used in each, phase 2A (interleave method) and 2B (uni-resident method) differ in the manner of utilizing the available disk work area.

During a particular pass, strings are read from one area, merged, and longer strings are written into the other area. During the succeeding pass, the input and output assignment of the two disk areas are reversed. As the strings are written on the disk, phase 2A employs a technique of interleaving the strings of each group. These strings are then merged in the following pass.

If phase 2B is used, one recording of the file may occupy almost the entire work area. Each time that a block of records is read into core storage, the next output block is written into the same area on the disk. This keeps the number of seeks at a minimum and permits the sorting of a larger file with a given work area.

With a given disk capacity for use as working storage, the maximum file size is almost twice as great if the uni-resident sort (phase 2B) is used.

Phase 2A or 2B is chosen automatically in phase 0 based upon a number of factors. Included among the factors which influence the selection are the following:

1. If two disk drives are available for the disk work areas, the interleave method (phase 2A) is generally more efficient. When the work area must be confined to only one drive, the uni-resident sort technique (phase 2B) is more efficient.
2. If the automatic checkpoint option is specified, the uni-resident sort (phase 2B) cannot be used.
3. If the disk output area overlaps part or all of the disk work area A, the uni-resident sort cannot be used.
4. If only one work area is specified, the uni-resident sort will be used.

PHASE 3 - FINAL SORT

Phase 3 performs the last pass of the external sort and provides the final output. This is usually accomplished in one pass, although in certain circumstances an extra pass may be required.

Final output may be to card punch, disk, or tape in combinations except those which include both disk and tape. Output to card punch, if specified, is provided as each record is moved to the final output area of core storage. Output to punched cards is limited to one card per record. Output to tape or disk is provided in the block sizes as specified by the OUTFIL statement.

IBM standard labels are processed for the output file if output is to tape with IBM standard labels or to disk.

Phase 3 provides exits 31 and 32 which may be activated to access a user-written routine. This may be employed to make format revisions, to update, delete, insert, translate, or summarize the output records, and/or to create "user" output labels.

PHASE 4 - MERGE ONLY

The merge-only phase can be used to merge existing pre-sorted files into one sequential file. A maximum of four tape, disk, or card input files can be merged; however, input cannot be from both tape and disk, nor can output be to tape and disk. Only one of the input files can come from cards.

The merge-only phase combines the input functions of phase 1 with the merge and output functions of phase 3. Since the input is presequenced, there is no necessity for the internal sort of phase 1 or the intermediate merge passes of phase 2. Phase 4 includes exits 41, 42, 43, and 44 which are comparable to exits in phases 1 and 3.

IBM standard labels are processed for the input file if input comes from tape with IBM standard labels, or from a disk input file; and IBM standard labels are processed for the output file, if output is to tape with IBM standard labels, or to disk.

PHASE 5 - TIMING PHASE

The timing phase calculates a set of timing estimates for sorting an input file with the given set of Job Control statements and Sort/Merge Control statements and with the _n1_ and _n2_ of the TIME operand in the OPTION statement. The _n1_ is the minimum file size, and all calculations are made to give a projection for that file size. The file size is then increased to (n1+n2) and a projection is made, provided the maximum file size is not exceeded. This continues until either a projection has been made for file size (n1+9n2) or the maximum file size has been exceeded. No actual sorting is done. The sorting technique uni-resident or variable-string interleave is identified and all estimates which are made are printed.

HOUSEKEEPING AREA AND WORK AREAS

A single-extent disk housekeeping area must be defined for all sorting and merging applications. If the Disk Sort/Merge Program is disk-resident, one cylinder must be reserved for the housekeeping area, starting and ending on cylinder boundaries. If card-resident, four cylinders must be reserved for the housekeeping area.

One or two disk work areas must be specified for sorting applications. If two work areas are used, they can be on the same disk drive or on separate drives. Each work area must be single-extent and must be large enough to contain the entire input file.

In specifying his disk work areas and output extents (if the output is to be written on disk), the user is primarily concerned with two factors:

1. That the disk areas which he specifies for work areas and for the output file are large enough to contain the file plus at least 4 extra tracks, but not so large as to waste file capacity.

2. That the areas specified are positioned relatively so as not to adversely affect sort efficiency.

The computation of the required disk work area depends upon whether the variable-string interleave (phase 2A) or the uni-resident (phase 2B) sorting technique will be used. The choice between methods is made automatically in phase 0; however, if he so desires, the user can usually force one choice or the other or determine in advance which technique will be used. (See <u>Phase 2 - External Sort</u>.)

DISK AREA CONSIDERATIONS FOR INTERLEAVE SORT

For the interleave sort, two disk work areas must be specified, preferably on two different drives. WORKA is to receive the output from phase 1. WORKA and WORKB are then used interchangeably as input and output during execution of phase 2.

During phase 1, the sort blocks resulting from the internal sort are written in WORKA. During the odd-numbered passes of phase 2, the strings are read from WORKA, merged, and the longer strings are written into WORKB . During the even-numbered passes of phase 2, the Disk Sort/Merge Program reads from WORKB, merges, and writes into WORKA.

If the output area should overlap WORKA, and if phase 3 is entered from an odd-numbered merge pass, then one additional pass is required to place the final output into the proper area. If the output area should overlap WORKB, and if phase 3 is entered from an even-numbered merge pass, then an additional pass is required. If the option FREEOUT is used, no additional pass is required in either case.

Figure 25 illustrates a variety of arrangements of disk storage areas for the interleave sort. Areas represented as adjacent may or may not be contiguous and may or may not be within the same pack. Any one area must be contained completely within one pack or (in the case of input and output areas) within packs mounted successively on the same drive. No consideration is given in the illustrations to possible variations in size requirements due to different blocking.

The cases illustrated in Figure 25 are:

CASE A10 -	Tape input and tape output
CASE A20 -	Disk input and tape output
CASE A21 -	Disk input and tape output with WORKB assigned to the same area as the input. This arrangement results in the destruction of the input but processes a larger file than CASE A20
CASE A30 -	Tape input and disk output
CASE A31 -	Tape input and disk output but with the output overlaying WORKB. This permits a larger file size than A30, but an extra copy pass may be required
CASE A32 -	Tape input and disk output. This arrangement is equivalent to CASE A31
CASE A40 -	Disk input and disk output
CASE A41 -	Disk input and disk output, but with output overlaying WORKB. This is equivalent to CASE A31
CASE A42 -	Disk input and disk output, but with output overlaying the input. This permits larger file size than A40, but results in destruction of the input
CASE A43 -	Disk input and disk output, but with output overlaying WORKA.
CASE A44 -	Disk input and disk output, but with WORKB overlaying input and output overlaying WORKB. Results in destruction of input and may necessitate a copy pass, but permits a larger file size
CASE A45 -	Disk input and disk output but with WORKB overlaying input and the output overlaying WORKA.

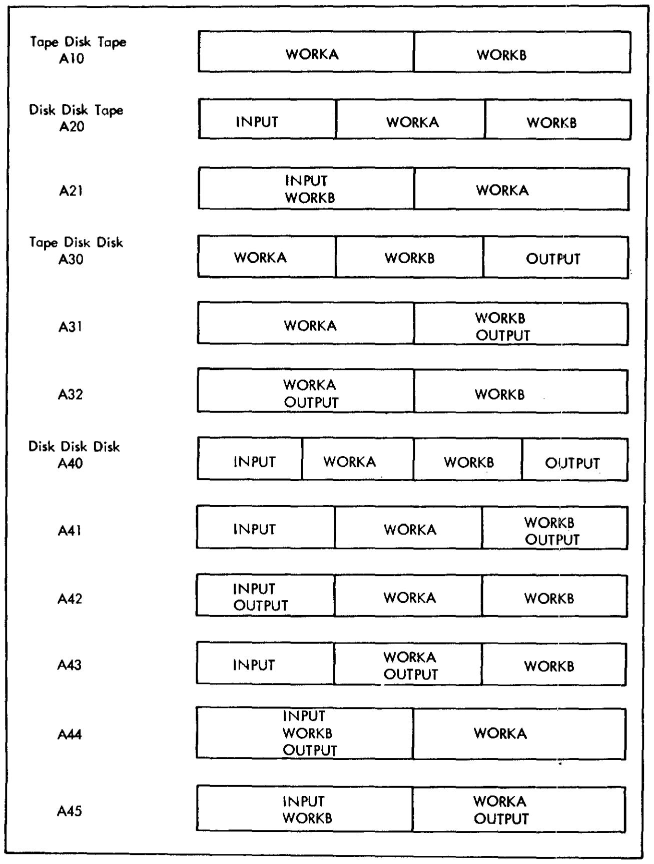


Figure 25. Arrangements of Disk Storage Areas for the Interleave Sort

Where different areas are shown in Figure 25 and in Figure 26, the different areas can be on the same or on different disk packs if two disk drives are used. The work areas must be single-extent, but the input and output areas can be single-extent or multi-extent (which includes multi-pack as long as all packs containing the input areas are mounted successively on one drive, and all packs containing the output areas are mounted successively on one drive). The Disk Sort/Merge Program (if disk-resident), the housekeeping area, and WORKA must be on-line at all times during execution.

In case A40, for example, all areas including the housekeeping area must be on one pack if only one disk drive is used. If two drives are used but only one pack per drive, then the areas can be distributed as desired, providing that each area is contained within one pack. If there are two drives, another possibility is to have the Disk Sort/Merge Program, the housekeeping area, and WORKA on one pack mounted on the first drive, and all of the other areas on one or more packs mounted successively on the second drive. For instance, the first (or only) pack containing INPUT is mounted first on the second drive followed by additional packs containing parts of INPUT if INPUT is multi-pack. After phase 1 has been completed, the pack containing WORKB is mounted (this could be the last pack used for INPUT or a separate pack) on the second drive. After phase 2 has been completed, the first (or only) pack which will contain OUTPUT is mounted on the second drive (this pack could be the pack containing WORKB or another pack). If OUTPUT is multi-pack, the packs are mounted in succession. The pack on the first drive remains mounted throughout execution.

DISK AREA CONSIDERATIONS FOR UNI-RESIDENT SORT

For the uni-resident sort, only one disk work area is used.

Figure 26 illustrates a variety of arrangements of disk storage areas for the uni-resident sort. Areas represented as adjacent may or may not be contiguous and may or may not be within one pack. The Disk Sort/Merge Program (if disk-resident), WORK, and housekeeping areas must remain mounted at all times during execution. The input or output area must be contained completely within one pack or within packs mounted successively on the same drive.

The cases illustrated in Figure 26 are:

CASE B10 -	Tape input and tape output
CASE B20 -	Disk input and tape output
CASE B30 -	Tape input and disk output
CASE B40 -	Disk input and disk output
CASE B42 -	Disk input and disk output but with output overlaying input area

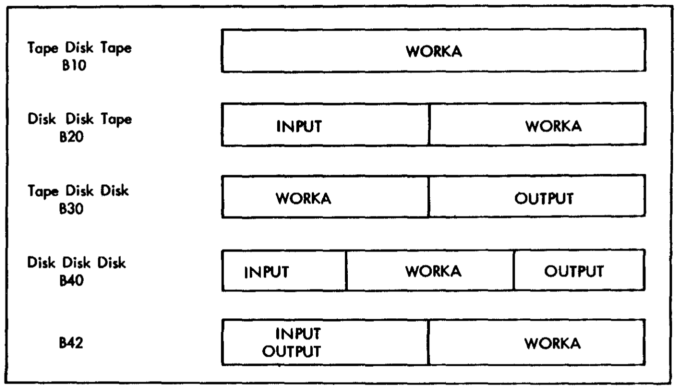


Figure 26. Arrangement of Disk Storage Areas for Uni-Resident Sort

Suppose a user has a customer file which needs to be updated frequently. The data is assembled, punched into cards, to be sorted to form an ordered file, and then to be merged with the master file. The following is one method of accomplishing this.

Assume that the user has a System/360 Model 20 with 16,384 bytes of core storage, a 2501 Card Reader, a 1442 Card Punch, a 1403 Printer, and two 2311 Disk Drives, Model 12. His Disk Sort/Merge Program is disk-resident for minimization of card handling for his sorting and merging needs. Assume further that the date is January 26, 1967, and all Job Control and Sort/Merge Control statements are to be printed.

To sort the data from the cards on disk, the user prepares the control statements shown in Figure 27.

The input file consists only of cards, but for output the sorted data will be stored on the disk file as well as being punched again. The punched output can be sent to another department for any needed action. The Disk Sort/Merge Program, the housekeeping area, the first work area, and the output file area are all assigned to the first disk drive. The second work area is assigned to the second disk drive. The output file area overlays the first work area.

```
0        1         2         3         4         5         6         7
1        0         0         0         0         0         0         0
// LOG
// JOB   SORT
// ASSGN SYS000,X'801',D4
// ASSGN SYS001,X'300',P2
// ASSGN SYS002,X'100',R4
// ASSGN SYS003,UA
// ASSGN SYS006,X'801',D4
// ASSGN SYS007,X'802',D4
// ASSGN SYS008,X'801',D4
// DATE  67026
// CONFG 001
// VOL   SYS000,SORTO
// DLAB  'WORK AREA                                   1PACK00',        C
               0001,67026,67026
// XTENT 1,000,0026000,0056009,'PACK00',SYS000
// VOL   SYS006,WORKA
// DLAB  'WORK AREA                                   1PACK00',        C
               0001,67001,67001
// XTENT 1,000,0026000,0056009,'PACK00',SYS006
// VOL   SYS007,WORKB
// DLAB  'SCRATCH AREA                                1PACK01',        C
               0001,67001,67001
// XTENT 1,000,0026000,0056009,'PACK01',SYS007
// VOL   SYS008,HSKPG
// DLAB  'HOUSEKEEPING AREA                           1PACK00',        C
               0001,67001,67001
// XTENT 1,000,0025000,0025009,'PACK00',SYS008
// EXEC
 SORT    FIELDS=(11,15,A,1,10,A,26,5,D,31,40,A),FORMAT=CH
 RECORD  TYPE=F,LENGTH=80
 INPFIL  BLKSIZE=(80,X)
 OUTFIL  BLKSIZE=2700
 OPTION  PRINT
 END
```

Figure 27. Sample Sort Application

```
0         1         2         3         4         5         6         7
1         0         0         0         0         0         0         0

// LOG
// JOB   SORT
// ASSGN SYS000,X'802',D4
// ASSGN SYS001,UA
// ASSGN SYS002,X'802',D4
// ASSGN SYS003,X'801',D4
// ASSGN SYS004,UA
// ASSGN SYS005,UA
// ASSGN SYS008,X'801',D4
// DATE  67026
// CONFG 001
// VOL   SYS000,SORTO
// DLAB  'MASTER FILE 193, 67026                     1MASTER',       C
               0001,67026,67116
// XTENT 1,000,0063000,0093009,'MASTER',SYS000
// VOL   SYS002,FILEA
// DLAB  'MASTER FILE 192, 67016                     1MASTER',       C
               0001,67016,67106
// XTENT 1,000,0032000,0062009,'MASTER',SYS002
// VOL   SYS003,FILEB
// DLAB  'WORK AREA                                  1PACK00',       C
               0001,67026,67026
// XTENT 1,000,0026000,0056009,'PACK00',SYS003
// VOL   SYS008,HSKPG
// DLAB  'HOUSEKEEPING AREA                          1PACK00',       C
               0001,67001,67001
// XTENT 1,000,0025000,0025009,'PACK00',SYS008
// EXEC
 MERGE   FIELDS=(11,15,A,1,10,A,26,5,D,31,40,A),FORMAT=CH,ORDER=2
 RECORD  TYPE=F,LENGTH=80
 INPLFIL BLKSIZE=(2700,X)
 OUTFIL  BLKSIZE=2700
 OPTION  PRINT
 END
```

Figure 28. Sample Merge Application

To merge the sorted data file just created with the master file, the user prepares the control statements shown in Figure 28. His input files are the newly created sorted file, which is on the first drive with the housekeeping area and the DISK Sort/Merge Program, and the old master file which is on the pack mounted on the second drive. The output file, or new master file, is written on the second drive so that both the old and the new master files are on the same pack. If there are several master files on the same pack, each new one could simply replace the oldest file on the pack.

Since the Disk Sort/Merge Program is disk-resident, only one cylinder is reserved for the housekeeping area. Merging does not require a work area.

The data cards themselves could have the following format (assuming the same format for each card):

Columns 1-10	First Name
Columns 11-25	Last Name
Columns 26-30	Date of Entry of this Card
Columns 31-70	Items purchased and sold
Columns 71-80	Balance Outstanding

If more than one card were required to enter all desired information in columns 31-70, then columns 1-30 would be duplicated in each successive card. Each card is considered to be a record.

During sorting, two records are placed in desired order by comparing data in the control fields of one record with data in the control fields of another record and then ordering the two records. Up to 12 control fields could be specified in order of importance, but in this case 4 fields are sufficient. The major control field in this case is the "last name" field, columns 11-25, and ascending sequence is desired. If data in the major control fields of two records compare unequally, no further comparisons are necessary. If they compare equally, then data in the second control field, the "first name" field (columns 1-10) would have to be compared and also placed in ascending order. If data compare unequally, no further comparisons are necessary. If they are equal, then the third fields, columns 26-30, with descending order specified, would have to be compared. If that comparison also is equal, then columns 31-70 with ascending order specified would be used as the fourth control field.

If data in columns 1-70 is identical in several records, in the merge operation, all those with equal control field data in File A will merge ahead of those with matching control data in File B, those in File B would merge ahead of File C, and those in File C ahead of File D. For each control field, the beginning position, length, and desired sequence must be given.

The format of the control fields is character or alphameric, and the records are 80 bytes long.

Data on the disk is blocked in units of 2700 bytes or 10 sectors. Each block contains 33 records with the last 60 bytes of each block used for other purposes. Each cylinder could contain 10 blocks or 330 records. For example, a file of 10,230 records would require 31 cylinders.

All control statement information as well as diagnostic messages, record counts, and end-of-phase messages are printed.

STANDARD LABELS

The Disk Sort/Merge Program processes standard IBM System/360 tape labels and standard Model 20 disk labels. Disk labels must be present and standard. "User" labels are not permitted on disk. The information required for checking the initial header label and creating the initial output header label for a disk file is provided by the DLAB and XTENT cards. (See Job Control Statements.) For a complete discussion of labels and label processing, see IBM System/360 Model 20, Disk Programming System, Control and Service Programs, Form C24-9006.

If standard tape labels are specified, one volume (VOL) label is the minimum that must appear on a volume (reel) of records, but as many as seven additional volume labels may be present; the additional volume labels will be bypassed and will never be overwritten. With IBM standard tape labels, the minimum for a file is one header (HDR) label and one trailer label (EOF or EOV). Up to seven additional header labels and up to seven additional trailer labels can be present on input; these will be read but not processed. Up to eight "user" header labels (UHL) and up to eight "user" trailer labels (UTL) can be present; these will be read but not checked or created; however, they can be checked or created with user-written routines. The "user" labels created by user-written routines will be written onto the output file. The information required for checking the initial input header label and for creating the initial output header label is supplied by the TPLAB card. (See Job Control Statements.)

A tape file with the minimum number of labels would be arranged as follows:

VOL 1	checked by the Sort/Merge Program
HDR 1	checked or created by the Sort/Merge Program
tape mark	
data	
tape mark	
EOF 1	checked or created by the Sort/Merge Program
tape mark	
tape mark	present only with EOV; omitted with EOF

The EOF would be replaced by EOV only if another volume were to follow before the file was completed.

A tape file with the maximum number of labels would be arranged as follows:

```
VOL 1     checked by the Sort/Merge Program

VOL 2     not checked or created
  .       by the Sort/Merge Program
  .       or by a user-
VOL 8     written routine

HDR 1     checked or created by the
          Sort/Merge Program

HDR 2     not checked or created
  .       by the Sort/Merge Program
  .       or by a user-
HDR 8     written routine

UHL 1     not checked or created
  .       by the Sort/Merge Program; can be
  .       checked by a user-written
  .       routine and will be written
  .       if created by a
UHL 8     user-written routine

tape mark
data
tape mark

EOF 1     checked or created by the
          Sort/Merge Program

EOF 2     not checked or created
  .       by the Sort/Merge Program or by
  .       a user-written routine,
  .       each EOF would be replaced
  .       by EOV if the end of the
EOF 8     file had not yet been reached

UTL 1     not checked or created
  .       by the program; can be
  .       checked by a user-written
  .       routine and will be written
  .       if created by
UTL 8     a user-written routine

tape mark
tape mark  present only with EOV, omitted
           with EOF
```

Disk Input

For disk input files the Sort/Merge Program verifies that the input pack(s) contains the specified input file within the extents and volumes specified.

Disk Output

For disk output files the Sort/Merge Program will perform the following functions:

- Verify that the disk pack(s) specified by the DLAB and XTENT card(s) is mounted on the output drive.

- Verify that the specified output areas of the output pack do not contain an unexpired file. If such a file is found, a message is printed followed by a stop giving the option of continuing as though the file had already expired.

- Delete the labels for any expired files whose extents conflict with the specified output extents.

- Create and write the output file standard disk label.

Disk Work Area and Housekeeping Area

For the specified disk work and housekeeping areas, the Disk Sort/Merge Program will perform the following functions:

- Verify that the disk pack specified by the XTENT card is mounted on the specified drive.

- Verify that the specified areas do not contain an unexpired file. If an unexpired file is found, a message is printed followed by a stop giving the option of continuing as though the file had already expired.

- Delete the label for any expired files whose extents conflict with the specified work areas or housekeeping area extent.

Tape Input

If standard labels are specified for tape input the Disk Sort/Merge Program will perform the following functions:

- Verify that the tape volume specified by the TPLAB card is mounted on the input drive assigned.

- Verify that the input volume contains the file which was specified.

- Read, but not check, user input headers, if any. These may be checked by the user's own routine if Exit 11 is activated.

- Check trailers for EOF or EOV condition.

- Read, but not check, user trailers, if any. These may be checked by a user subroutine through the same exit used to check user headers.

- For multi-reel input, perform these functions for each reel and verify that the reels are in the proper sequence.

  Reels are processed until either an EOF trailer label is read or as many reels have been processed as were specified in a VOLUME entry, if any, whichever comes first. (See INPFIL Statement.)

Tape Output

If standard labels are specified for tape output, the Disk Sort/Merge Program will perform the following functions:

- Verify that the tape volume specified by the TPLAB card is mounted on the output drive assigned.

- Verify that the reel mounted on the output drive does not contain an unexpired file. If such a file is found, a message is printed followed by a stop with the option of continuing and causing the label to be overwritten and the file to be used as though the file had already expired.

- Create and write the first output header after the existing volume label.

- Write user output headers if these are created by a user subroutine. (See Exit 31.)

- Write a tape mark following the last header label.

- Create and write a trailer label preceded by a tape mark at the end-of-reel or end-of-file.

- Write user output trailers if these are created by a user subroutine. (See Exit 31.)

- Write a tape mark following the last label.

- For multi-reel output, perform the same functions for each reel.

NON-STANDARD TAPE LABELS AND UNLABELLED TAPE FILES

Although programmed exits are provided for the processing of user tape labels, the processing of non-standard labels is not allowed. Input tapes containing non-standard labels can be accepted, but the labels themselves will be bypassed. If non-standard labels are present, this must be specified in the OPTION card so that they may be bypassed. (See OPTION Statement.) If non-standard labels do exist they must be followed by a tape mark. Non-standard labels cannot be written on the output tape file.

Unlabelled tape files can be processed if this is specified in the OPTION card. On input, the first record read from an unlabelled input volume can be either a tape mark or a data record; if it is not a tape mark, it will be treated as the first input block.

With unlabelled input tape files or input tape files with non-standard labels, as many volumes are read as are specified in the VOLUME entry of the INPFIL statement if that entry is given. If the VOLUME entry is not given, one volume is read.

A tape mark is normally written before the first data block on output. If the output tape file is unlabelled, the tape mark can be omitted if the no-tape-mark (NOTPMK) entry is given in the OPTION Statement. (See OPTION Statement.)

Provision is made in phase 1, phase 3, and phase 4 for the addition of subroutines written by the user.

A modification should be added to the Disk Sort/Merge Program only if careful system analysis indicates that it is more economical than the adoption of alternative procedures, such as making equivalent modifications to another program in the system, or even writing a special pre-edit or post-edit program to perform the desired functions. Some factors to be considered before adding a modification are:

- The reservation of core storage for additional programming may restrict the normal assignment of data areas to the extent of increasing the total Sort/Merge time for a given application.
- A modification subroutine added to the Disk Sort/Merge Program must, in most cases, include instructions to save and restore registers.
- Other programs in the customer's library may be more readily modified, as well as being better known to the programmer.

INTERFACE WITH USER-WRITTEN ROUTINES

The Disk Sort/Merge Program links to the user-written routines through the use of branch instructions (BAS) called exits. The following exits exist:

```
Phase 1 - Exit 11, Exit 12, Exit 13
Phase 3 - Exit 31, Exit 32
Phase 4 - Exit 41, Exit 42, Exit 43,
          Exit 44
```

To activate one or more exits, the user must make the required entries in the MODS control statement (see <u>MODS Statement</u>). He must also assemble a branch table at the origin of the user-written routine for each phase. The origin of the branch table for each phase must always be a halfword boundary. It is developed by subtracting the total size of user-written routines for that phase from the total core storage capacity available and then adjusting to the next lower halfword boundary.

```
Example 1:  CONFG 001             16384 bytes
            subtract 2               -2
                                  -----
                                  16382
subtract user-program length       -554
                                  -----
                                  15828
```

The origin address for the user-program is thus 15828.

```
Example 2:  CONFG 001             16384 bytes

but the OPTION statement specifies
                                STORAGE=15575
truncate to a half-word boundary           -1
                                        -----
                                        15574
              subtract 2                   -2
                                        -----
                                        15572
subtract user-program length             -554
                                        -----
                                        15018
```

Thus the origin address for the user-program is 15018.

Card-Resident User Program

If the user program is card-resident, it will be loaded starting at the core storage address computed as follows:

$$\text{Load address} = \left\lfloor \frac{C}{2} \right\rfloor * 2 - 2 - \left\lceil \frac{L}{2} \right\rceil * 2$$

L = user program length as specified in the MODS statement
C = 16384 for CONFG 001 (See CONFG statement)
C = 12288 for CONFG 101
C = the value stated in the STORAGE operand if that operand is specified

$\lfloor\ \rfloor$ = Rounded down to whole number
$\lceil\ \rceil$ = Rounded up to whole number

If the load address differs from the origin address of the user program, any address constants contained within the user program will be modified accordingly.

Disk-Resident User Program

If the name is present within the PH1, PH3, or PH4 operand of the MODS statement, a disk-resident user program is indicated. For such a program, a minimum load address is computed using the same formula as for the card-resident user programs. This address is then compared to the actual address as specified for the program in the core-image directory. If the computed address is higher, then the input will reject with an appropriate error message;

otherwise, the program is loaded at the actual address as specified in the core-image directory. Since the program is not relocated, no address constants are altered.

Each time the Disk Sort/Merge Program reaches an exit which has been activated, it transfers control to a particular position of the user's branch table, which in turn branches to the proper user subroutine within the user-written routine. Thus the branch table serves as a set of pivot points, one for each exit and its associated subroutine. The last instruction of each subroutine must branch back to an instruction of the main program immediately following the exit.

The two general registers involved in the exits are registers 14 and 15. The branches function in the following manner:

The exit instruction is BAS 14,X(0,15). This instruction stores the address of the next sequential instruction in register 14, and the branch address is formed from the constant base address stored in register 15 plus X, which is equal either to zero or a multiple of 4. Register 15, which is initialized by the Disk Sort/Merge Program, must be specified as the base register in the USING statement in all user-assembled subroutines. This is done by inserting the statement "USING *,15" before the first branch table entry when assembling the user routine for a phase.

If the user chooses to activate one or more exits in any phase, he must prepare, immediately ahead of his routine, a branch table that includes a 4-byte entry for each branch up to and including the last exit to be used in that phase. The proper branch table entry for those exits which are not used is "BC 15,0(14)". If, for example, he chooses to use exit 32 but not exit 31, he should include a "BC 15,0(14)" as a dummy entry for exit 31. The entries for the unused exits that follow can be omitted. The label of the first instruction of his subroutine might be LAB2, although he is free to assign any label he chooses. For the example just cited the branch table would contain:

BC 15,0(0,14)	= Dummy branch table entry for exit 31
BC 15,LAB2(0,15)	= Branch table entry for exit 32

It is advantageous to the user to restrict his subroutines to the smallest possible core storage requirement, as any core storage taken up by his routines reduces the size of input/output and internal sorting areas and diminishes correspondingly the throughput rate. In any event, the user-written routines must not reduce the total capacity available to the Disk Sort/Merge Program below 8192 bytes, nor may a disk-resident user-program exceed 4050 bytes for any one phase.

The user-written routine must save the contents of any register which is altered, and the contents must be restored before returning to the main program unless otherwise specified in the instructions for that paticular exit.

In a user-written routine, any card device not needed by the Disk Sort/Merge Program can be used for input or output. If this device is used for reading control statements, the control statement cards must be separated from the data to be read by the user-written routine by a blank card, or a /*b card.

All input/output routines for devices used by a user-written routine must be completely self-contained in the user-written routine.

The user-written routines will be automatically loaded from cards or disk and cannot contain their own loaders.

EXPLANATION OF PROGRAM EXITS

Exit 11 (E11)

This exit is available during phase 1 and can be used only with tape with IBM standard labels. It can be used to check user tape input header and trailer labels. The Disk Sort/Merge Program will have read the user label (UHL or UTL) before branching from the exit into the user-written routine. The core storage address of the first byte of the 80 character label will be in register 13.

The byte in core storage immediately following the exit instruction contains a code to indicate the label environments. This byte can be tested for X'01', X'02', or X'04' to indicate an OPEN, EOV, or an EOF condition, respectively.

The reentry to the main program must bypass the halfword containing the environment byte. The reentry instruction would be:

```
BC 15,2(0,14)
```

The contents of any register altered by the user program must be restored before reentry to the main program.

The user routine will be entered for each user header label (UHL) or user trailer label (UTL) on the tape until an end-of-file condition is detected.

Exit 12 (E12)

This exit is available in phase 1 each time a block of data is read into the input area from the input file. It can be used for translation of data not already in an acceptable format. For example, if the input file were a tape file consisting of variable-length records, this exit could be used to convert these to fixed-length and to reformat the block accordingly.

When the user routine is entered through exit 12, register 12 contains the number of bytes in the block just read. This exit occurs immediately after the block has been read and before any action has been taken by the program. If the user program alters the block length, register 12 must be updated accordingly. The input block length which is specified in the INPFIL card must be the greatest block length which may exist before or after the exits (for disk input it is not permissible to expand block lengths beyond the next higher multiple of 270 bytes).

The address of the leftmost position of the input block is in register 13 when exit 12 is used.

In addition to the reformatting of input blocks, exit 12 can also be used to perform operations on the individual records. For example, control fields may be translated (the user must provide any translation tables which may be required) or records may be deleted. The DELBLANK option, if specified, is invoked following exit 12; thus deletion could be accomplished by changing the character in the position of the blank record indicator to the deletion character specified in the DELBLANK option (see RECORD Statement).

Except as noted above for register 12, the contents of any register altered by the user program must be restored. The following instruction should be used for re-entry to the main program:

```
BCR 15,14
```

Exit 13 (E13)

This exit is available during phase 1 if a tape or disk input block should be unreadable. Exit 13 can be used to print the data in error.

Register 12 will contain the address of the read CCW. For tape input, the core address of the data block in error will occupy bytes 3 and 4 of the CCW; the length of the data block will occupy bytes 5 and 6. For disk input, the sector count is in byte 2 of the CCW, the core address is in bytes 3 and 4, and the count field core storage address is in bytes 5 and 6.

If, during execution of the user-written routine, it is decided that the data may be accepted, phase 1 should be re-entered by a BC 15,4(0,14) instruction. If the data cannot be accepted, phase 1 must be re-entered by a BC 15,0(0,14) instruction. In the latter case, the Disk Sort/Merge Program will either bypass the data or, if the BYPASS option was not specified, terminate the job.

Exit 31 (E31)

This exit is available during phase 3 and can only be used with tape output with IBM standard labels. It can be used to create user tape header and trailer labels. When the exit occurs, the address of the leftmost position of an 80 character buffer which can be used to construct the label is in register 13.

The byte in core storage immediately following the exit instruction contains a code to indicate the label environment. This byte can be tested for X'11', X'12', or X'14' to indicate an OPEN, EOV or an EOF condition, respectively.

In order to cause a label which it has created to be written, the user routine must re-enter the main program through instruction BC 15,2(0,14). After writing the label the main program will again return control to the user routine through exit 31. When there is no label to be written, the user routine must re-enter the main program through instruction BC 15,6(0,14).

A tape mark will be written following the last label.

The contents of any register altered by the user routine must be restored before re-entry to the main program at either address.

Exit 32 (E32)

This exit occurs for each record after it has been moved to the output area in the last pass of the sort. It can be used for insertion, deletion, alteration, or summarization of records, or for data translation. When the exit occurs, the address of the record just moved is available in register 13.

The byte in core storage immediately following the exit instruction serves as a switch to identify the last record of the file. This byte, which normally contains

X'00', will contain X'01' when the last record has been moved to the output area. The halfword containing this byte must be bypassed when re-entering the main program.

If it is desired to delete the record which has just been moved to the output area prior to the exit, the user program must reenter the main program through instruction BC 15,2(0,14). After deleting the record, the main program will again return control to the user program through exit 32. It would, of course, be an error for the user program to reenter the main program at the deletion point a second time consecutively.

If it is desired to insert a record following the last record just moved to the output, or in the place of a record just deleted, the following actions would be required by the user program:

1. Create the record to be inserted in the user's own program area.
2. Place the address of the record to be inserted in register 12.
3. Reenter the main program through instruction BC 15,6(0,14).

In any case, the record to be inserted cannot be longer than the length specified by length3.

After inserting the record, the main program will again return control to the user program through exit 32. When it is desired to re-enter the main program without deleting or inserting, the user program should branch back via a BC 15,10(0,14) instruction.

If record lengths are altered by the user program, all records can be altered from length2 to length3. (See <u>RECORD Statement.</u>) If length2 differs from length3, but the user program does not perform the alteration, the records will be automatically truncated to length3.

The contents of any register altered by the user program except R12 must be restored before re-entry to the main program at any of the three re-entry points. The original contents of R12 must be restored before the final re-entry.

Exit 41 (E41)

This exit is available during the merge phase of a merge-only operation and can be used only with tape with IBM standard labels. It can be used to check user tape input header and trailer labels or to create user output header or trailer labels. Thus, this exit performs the same functions for merge-only operations as exits 11 and 31 combined. When the exit occurs, the address of the left-most position of the label area is in R13.

The byte in core storage immediately following the exit instruction contains a code to indicate the label environment. This byte can be tested for X'01', X'02', X'04', X'11', X'12', or X'14' to indicate input OPEN, input EOV, input EOF, output OPEN, output EOV, or output EOF, respectively. Files are opened in the following order:

SYS000, SYS002, SYS003, SYS004, SYS005

For input conditions, the main program must be re-entered through instruction BC 15,2(0,14). For output conditions, the program is re-entered through instruction BC 15,2(0,14) to cause a label to be written. After writing the label, the main program will again return control to the user program through exit 41. When there is no label to be written for an output condition, the user program must re-enter the main program through instruction BC 15,6(0,14).

The contents of any register altered by the user program must be restored before re-entry to the main program at either address.

Exit 42 (E42)

This exit is available during the merge phase of a merge-only operation. Its function is identical to that of exit 12 (see <u>Exit 12</u>).

Exit 43 (E43)

This exit is available during the merge phase of a merge-only operation. Its function is identical to that of exit 13 (see <u>Exit 13</u>).

Exit 44 (E44)

This exit is available during the merge phase of a merge-only operation. Its function is identical to that of exit 32 (see <u>Exit 32</u>).

APPENDIX D - DIAGNOSTIC MESSAGES

The following diagnostic messages, preceded by the word ERROR, may be printed. A halt occurs when the diagnostic messages have been printed. A restart is not possible except where "Operator choice" is stated. (The operator may choose to continue or terminate the job.)

Diagnostic Message	Meaning
30019	The first 15 columns of the continuation card are not blank or the statement does not resume in column 16.
30020	Required Statement operand(s) have been omitted.
30021	One of the following: 1. Numeric field contains non-numeric characters. 2. The numeric value exceeds the permitted maximum.
30022	A card in the Sort-Merge statement deck is not blank in column 1.
30023	Operation code of Sort/Merge Control statement does not start within the first 15 columns.
30024	Operation code of Sort/Merge Control statement is invalid.
30025	An operand of the Sort/Merge control statement is not recognizable, or an essential character (equal sign or left parenthesis) is missing, or the operand appears in duplicate.
30026	Control field position in FIELDS operand of SORT statement is 0.
30027	No comma after the control field position or the length specification in FIELDS operand of SORT statement.
30028	Number of bytes specified for any one control field is 0 or greater than 256.
30029	Either of the following: 1. FIELDS operand of SORT statement does not contain a sequencing code. 2. Code used to indicate the direction of sequencing is invalid.
30030	More than 12 control fields are specified in one FIELDS operand of the SORT statement.
30031	Control field in FIELDS operand of a SORT or MERGE statement is not followed by a comma or a close parenthesis.
30032	An operand in the SORT or MERGE statement is followed by a character other than a comma or blank.
30033	Invalid specification in FORMAT operand of SORT statement.
30034	An operand in the RECORD statement is not recognizable, or an essential character (equal sign or left parenthesis) is missing, or the operand appears in duplicate.
30035	Blank-record indicator position in DELBLANK operand is 0.
30036	Comparison character in DELBLANK operand is not followed by closing parenthesis.
30037	Operand in RECORD statement is followed by a character other than a blank or a comma.
30038	More than three lengths specified in LENGTH operand of RECORD statement.

30039	Last (or only) length value in LENGTH operand is not followed by closing parenthesis.
30040	An operand in the INPFIL statement is not recognizable, or an essential character (equal sign or left parenthesis) is missing, or the operand appears in duplicate.
30041	BLKSIZE operand does not contain the fixed-length block specification.
30042	More than four volume specifications in the VOLUME operand.
30043	The specification for the VOLUME or the BLKSIZE operand in the INPFIL statement is preceded by an open parenthesis but is not followed by a close parenthesis.
30044	Operand in INPFIL statement is followed by a character other than a blank or a comma.
30045	An operand in the OUTFIL statement is not recognizable, or an essential equal sign or associated specification is either missing or not valid, or the operand appears in duplicate.
30046	Operand in OUTFIL statement is followed by a character other than a blank or a comma.
30047	An operand in the OPTION statement is not recognizable, or an essential character (equal sign or left parenthesis) is missing, or the operand appears in duplicate.
30048	OPTION operand STORAGE specifies either: 1. A value in excess of the available capacity specified in the job-control CONFG statement or 2. A value less than 12286.
30049	TIME operand specifies either of the following: 1. A value for minimum file size (n1) of less than four digits or more than five digits. 2. A value for the file size increment (n2) of less than three digits or more than five digits.
30050	Tape label type specification for input file (in LABEL operand) is other than N, S, or U.
30051	Tape label type specification for output file (in LABEL operand) is other than S or U.
30052	Tape label type specification for input file (in LABEL operand) is not followed by a comma.
30053	Tape label type specification is preceded by an open parenthesis, but is not followed by a close parenthesis.
30054	Operand in OPTION statement is followed by a character other than a blank or a comma.
30056	An operand in the MODS statement is not recognizable, or an essential character (equal sign or left parenthesis) is missing, or the operand appears in duplicate.
30057	The PH1 operand appears in duplicate, or in combination with the operand PH4.
30058	The name of the user-written routine in the PH1 operand is longer than six characters, or is not followed by a comma.
30059	Exit specification in PH1 operand has been omitted.
30060	Exit specification in PH1 and/or PH3 field operand is invalid.

30061	The PH3 operand appears in duplicate, or in combination with PH4.
30062	The name of the user-written routine in the PH3 operand exceeds six characters, or is not followed by a comma.
30063	Exit specification in PH3 operand has been omitted.
30064	MODS statement operand is not followed by a closing parenthesis.
30065	Closing parenthesis at end of MODS-statement operand is not followed by a comma or a blank.
30066	Duplicate Sort/Merge Control statements.
30067	Neither sort nor merge operation specified.
30068	Either of the following: 1. Specification of PH4 user program for sort. 2. Specification of PH1 or PH3 user program for merge.
30069	Any of the following: 1. Order of merge specified as greater than 4. 2. Order of merge specified as less than 1. 3. Order of merge specified for sort. 4. Order of merge not specified.
30070	TYPE=F not specified in the RECORD statement.
30090	The MODS statement contains a duplicate PH4 operand or PH4 in combination with PH1 or PH3.
30091	The PH4 user program name exceeds 6 characters or is not followed by a comma.
30092	Although PH4 is specified in the MODS statement, no exit is specified.
30093	There is an unidentifiable exit specification for PH4.
30101	The Sort/Merge control statement cards (or the user program deck, if used) are not followed by an EOF card, a blank card, or a last card condition.
30102	The specified user program name is not present in the core-image directory.
30103	The origin address for the disk-resident user program as specified in the core image directory is lower than the origin address computed using the length value in the MODS statement.
30104	A disk-resident user program is larger than 4050 bytes.
30105	The card-resident user program is larger than specified in the MODS statement.
30106	The card-resident user program would reduce working core storage capacity below 8K.
30107	A card in the user program deck does not have the proper load card code (Name) in column 1.
30108	The first card of the user program deck is not an ESD card.
30201	More than one control field format has been specified.
30203	VERIFY specified in OPTION statement, but output device is other than disk.
30204	ADDROUT specification is invalid because: 1. Input is other than disk, or 2. Input consists of multiple volumes, or 3. Merge has been specified.

30206	NOTPMK specified in OPTION statement, but the label type specification for tape output is other than U (unlabeled)
30207	FREEOUT specification in OUTFIL statement is invalid because: 1. SYS007 has not been assigned, or 2. SYS000 is assigned, or 3. Merge has been specified.
30208	The control field extends beyond the end of the record.
30209	Control field for zoned decimal or packed decimal data exceeds 16 bytes.
30210	No control field specification in the SORT or MERGE statement.
30211	ADDOUT is specified, but the total control field length plus eight bytes, exceeds the record length.
30212	Either of the following: 1. Length 3 in LENGTH operand of RECORD statement specifies a value higher than the one specified for length 2. 2. Length 2 or length 3 specifies a value higher than the value specified for length 1.
30213	Specified output record-length exceeds specified output block-length.
30214	Specified value for the blank record indicator position in DELBLANK operand is higher than the value specified for record length.
30215	Specified length of input records plus the number of bytes to be skipped (if so specified in the INPFIL operand SKIPBYTE) exceeds the specified input block-length.
30216	Statement operands specify tape output functions, but ASSGN statement for tape output device has been omitted.
30218	Specified length of user-written routine does not accommodate the branch table.
30219	Statement operands specify tape input functions, but ASSGN statement for tape input device has been omitted.
30220	Extents specified for SYS002 and SYS006 or for SYS006 and SYS007 overlap.
30221	Either of the following: 1. Output extent overlaps WORKA extent and no WORKB extent has been specified. 2. Output extent overlaps both WORKA and WORKB extents.
30222	SORT cannot be permormed because: 1. The extent for WORKA is overlapped by the input or output extent and no extent has been specified for WORKB, or 2. CKPT has been specified, but no extent has been specified for WORKB.
30223	Available core storage capacity is insufficient to perform the sort job with the input and output block sizes, record lengths, and options specified.
30224	Total number of bytes specified for the control field(s) exceeds 256.
30225	Either of the following: 1. Specified length of records is 0. 2. RECORD operand LENGTH has been omitted.
30226	One of the following: 1. Specified block size in INPFIL statement is 0. 2. INPFIL operand BLKSIZE has been omitted. 3. Input record length exceeds input block length.

30227	One of the following: 1. Specified block size in OUTFIL statement is 0. 2. OUTFIL operand BLKSIZE has been omitted. 3. Output record length exceeds output block length.
30228	Either of the following: 1. SYS006 has been assigned to an I/O device other than disk. 2. SYS006 has not been assigned.
30229	Symbolic input device is assigned to a disk drive, but no appropriate extent has been provided.
30230	One of the following: 1. I/O device specified for SYS002 is not a valid input device. 2. SYS002 has not been assigned. 3. Two symbolic devices have been assigned as card input. 4. Fewer input devices than files have been specified in the order of merge for a merge application.
30231	Either of the following: 1. SYS007 has been assigned to a device other than disk. 2. No extent has been provided for SYS007.
30232	Symbolic output device is assigned to a disk drive, but no appropriate extent has been provided.
30233	One of the following: 1. SYS000 (or SYS001) has been assigned to a non-valid output device. 2. SYS000 has not been assigned. 3. Both SYS000 and SYS001 have been assigned to card devices.
30234	File size specified in the OPTION operand TIME is less than 1000.
30235	File-size increment specified in the OPTION operand TIME is less than 100.
30238	The specified work area is too small to accommodate the number of records specified as the minimum file size in OPTION operand TIME.
30250	SYS000 not assigned.
30253	CKPT, RESTART, or TIME has been specified for a merge application.
30254	In a sort application, volumes for more than one tape file have been specified. In a merge application, volumes have been specified for a number of files greater than the specified order or merge.
30255	A record length of less than 3 bytes has been specified.
30256	One card or tape device has been specified as both an input and output device in a merge application.
30258	A function has been specified which is not included in this version of the program.
30259	SKIPBYTE specified in combination with Merge or user exit E12.
30301	Housekeeping entent not provided,
30302	Housekeeping extent is too small. (Four cylinders are required for card-resident system; one cylinder for a disk-resident system, beginning at a cylinder boundary, SYS008 cannot be multi-extent.)
30303	Wrong volume mounted on SYS008.
30305	The specified file label cannot be created because of a conflicting extent with an unexpired file.

30306	The existing label for the input file contains a file serial number not the same as that in the DLAB card. Operator choice.
30307	The existing label for the input file contains a volume sequence number not the same as that in the DLAB card. Operator choice.
30308	The existing label for the input file contains a creation date unequal to that in the DLAB card. Operator choice.
30309	The existing label for the input file has an expiration date unequal to that in the DLAB card. Operator choice.
30310	The specified input file is identified by its label as an indexed-sequential file.
30311	The extent specified in the input data is not covered by an extent in the file label.
30312	There is no sector available in the VTOC for creating the new label.
30313	A label with invalid format specification exists within the VTOC.
30314	Extent count in an existing label is incorrect.
30316	Invalid storage-capacity specification in CONFG statement.
30317	Card-resident object deck for the Sort/Merge program contains an invalid or non-applicable card.
30318	Specified input label is not present.
30319	Required file cannot be opened because the label is not present in the job-control statements.
30320	More than 15 extents are specified for the input or output file.
30321	The required volume is not mounted on the specified drive. (This follows a message instructing the operator to mount the required volume.) Operator choice.
30322	The input or output file contains more than 15 extents.
30323	The Format-4 VTOC label is not present at the file address specified in the volume label.
30324	The specified file cannot be opened because of a conflicting extent. Operator choice.
30325	Zoned Decimal, Packed Decimal, or Fixed-point Integer control fields have been specified, but these fields overlap.
30326	Input record length is specified as greater than 80 for card input.
30327	Housekeeping unit (SYS008) is unassigned.
30328	More than one extent has been specified for SYS006, SYS007, or SYS008.
30329	The label for the specified file cannot be created because of a duplicate data set in the VTOC. Operator choice.
30330	Job Control data contains two or more files which are assigned to a common disk drive, and which must be accessed during a common phase of the program, and at least one of which would require the mounting of a different file serial number than the other files or because the file is multi-pack.
30331	The extent is not large enough to contain a complete block, or a work extent is not large enough to contain the first string group.

Name and Operation Code	Operands	Function	When Required
// ƀJOB	SORT	Initializes Sort/Merge run	Always
// ƀASSGN	SYSxxx,X'cuu', dd,X'ss'	Physical device assignment of each symbolic unit (SYSxxx) used	Always unless still in core storage from a previous job
// ƀASSGN	SYSxxx, UA	For making a symbolic unit unassigned	
// ƀDATE	yyddd	Used in processing standard labels	
// ƀCONFG	xxx	Available core storage	
// ƀVOL	SYS000, SORTO	Output file for sort or merge	Always required for disk files or for tape files with IBM standard labels
	SYS002, SORTI	Input file to a sort operation	
	SYS002, FILEA	First merge input file	
	SYS003, FILEB	Second merge input file	
	SYS004, FILEC	Third merge input file	
	SYS005, FILED	Fourth merge input file	
	SYS006, WORKA	Disk work area	for sort
	SYS007, WORKB	Second disk work area	for sort, if 2 work areas are specified
	SYS008, HSKPG	Housekeeping disk area	Always
// ƀTPLAB	'aa...aa'	With each VOL for tape with IBM standard labels	if tape contains IBM standard labels
// ƀDLAB	'aa...aa' c ssss,yyddd,yyddd	With each VOL for disk	for each disk file
// ƀXTENT	1,sss,cccchhh,cccchhh,'abcdef',SYSxxx	One or more with each DLAB card	for each disk file
// ƀLOG		Logs all Job Control statements	Optional
// ƀNOLOG		Stops logging of Job Control statements	Optional
// ƀFILES	SYSxxx,n	Position tape files	Optional
// ƀEXEC		Physically last Job Control statement	Always

Figure 29. Summary of Job Control Statements

Area	SYSxxx	Disk Only	Disk and Card	Tape Only	Tape and Card	Card Only
Output	SYS000	Disk or UA (UA only with FREEOUT)	Disk	Primary Unit	Tape	Card
	SYS001	UA	Card	Alternate unit or UA	Card	UA
Sort Input	SYS002	Disk	Disk	Primary Unit	Tape	Card
	SYS003	UA	Card	Alternate unit or UA	Card	UA
Merge Input (SYS002 has top merging priority, then SYS003, etc.) (At most one input file can be from cards)	SYS002	Disk	Disk or Card	Tape	Tape or Card	Card
	SYS003	Disk or UA	Disk or Card	Tape or UA	Tape or Card	UA
	SYS004	Disk or UA	Disk or Card or UA	Tape or UA	Tape or Card or UA	UA
	SYS005	Disk or UA	Disk or Card or UA	Tape or UA	Tape or Card or UA	UA
Sort Work Areas	SYS006	Disk				
	SYS007	Disk or UA				
Housekeeping Area (required)	SYS008	Disk				

Figure 30. Summary of Device Assignments

Operation Code	Operands and Associated Values	If Req	If used, only with	Default Values
SORT	FIELDS=(p1,m1,s1,....,p12,m12,s12)	Req	sort	
	FORMAT=BI or CH or ZD or PD or FI		sort	BI
	CKPT		sort	
MERGE	FIELDS=(p1,m1,s1,...,p12,m12,s12)	Req	merge	
	FORMAT=BI or CH or ZD or PD or FI		merge	BI
	ORDER=1 or 2 or 3 or 4	Req	merge	
RECORD	TYPE=F	Req		
	LENGTH=(1, 2, 3)	Req		
	DELBLANK=(p,q)		input	q=blank
INPFIL	BLKSIZE=(n,X)	Req		
	BYPASS			
	OPEN=RWD or NORWD		tape	RWD
	CLOSE=RWD or NORWD or UNLD		tape	RWD
	VOLUME=n or (na,nb,nc,nd)		tape	
	SKIPBYTE=n		disk and tape input	
	PRESEQ			
OUTFIL	BLKSIZE=n	Req		
	OPEN=RWD or NORWD		tape	RWD
	CLOSE=RWD or UNLD or NORWD		tape	RWD
	FREEOUT		disk sort	
MODS	PH1=(name,length,E11,E12,E13)		sort	
	PH3=(name,length,E31,E32)		sort	
	PH4=(name,length,E41,E42,E43,E44)		merge	
OPTION	PRINT			
	STORAGE=n			
	LABEL=(i,o) where i=S, U, or N o=S or U		tape	
	ADDROUT		disk	
	VERIFY		disk	
	NOTPMK		tape	
	RESTART		sort	
	TIME=(n1,n2)		sort	
END		Req		

Figure 31. Summary of Sort/Merge Control Statements

<u>Addresses Out (ADDROUT):</u> Sort whose output file consists of the disk file addresses of the sorted records rather than the records themselves.

<u>Alternate Tape Drive:</u> When two drives are given for one multi-reel tape file, the first drive is the primary drive and the second drive is the alternate drive. Reels are mounted such that the first is on the primary drive, the second reel on the alternate drive, the third on the primary drive, etc. If no alternate drive is given, all reels must be read or written on the primary drive.

<u>Ascending Order:</u> A sequence of records such that the control fields of each successive record collate equal to or higher than those of the preceding record.

<u>Ascending Sequence:</u> Ascending order.

<u>ASCII:</u> American Standard Code for Information Interchange.
<u>b:</u> Symbol for a blank space.

<u>Block:</u> Can be
1. A physical group of records grouped for the purpose of conserving tape or disk storage space or increasing the efficiency of access or processing.

2. A group of bytes transferred to or from a physical unit device in one operation; may contain one or more logical records.

<u>Byte:</u> The basic unit of information in the System/360. Every byte consists of 8 bits, each having a value of zero or one.

<u>Character:</u> An 8-bit (1-byte) code that can be manipulated in the core storage of the central processing unit (CPU).

<u>Checkpoint:</u> A point in a program about which sufficient information is stored to permit restarting the problem program from that point.

<u>Checkpoint Records:</u> Records that contain the status of the job and the system at the time the records are written by the checkpoint routine. These records provide the necessary information for restarting a job without having to return to the beginning of the job.

<u>Checkpoint/Restart:</u> A means of restarting execution of a program at some point other than the beginning. When it is necessary to restart a program at a point other than the beginning, the restart procedure uses the checkpoint record to reinitialize the system.

<u>Collating Sequence:</u> The relative order of characters upon which a sort or merge is based.

<u>Control Field:</u> A group of contiguous bytes that are within a data record. The sort or merge of the records is based on the collating sequence as applied to these bytes.

<u>Data File:</u> A collection of related records organized in a specific manner. For example, a payroll file (one record for each employee, showing his rate of pay, deductions, etc.) or an inventory file (one record for each inventory item, showing the cost, selling price, number in a stock, etc.).

<u>Default Value:</u> Value assumed when the value is not given in a control statement.

<u>Descending Order:</u> A sequence of records such that the control fields of each successive record collate equal to or lower than those of the preceding record.

<u>Descending Sequence:</u> Same as Descending Order.

<u>Extent:</u> Area of a disk file specified by an upper limit and a lower limit.

<u>EBCDIC:</u> (Extended Binary Coded Decimal Interchange Code) A specific set of 8-bit codes standard throughout System/360. Mnemonic term on a control card that defines a function to be performed or identifies a value being specified.

<u>File:</u> The same as Data File.

<u>File Label:</u> Label containing information applicable to a given data file or portion of a data file stored on a particular volume.

<u>Fixed-Length Record:</u> A record having the same length as all other records with which it is logically or physically associated.

<u>Fixed-Point Integer:</u> Storage technique whereby one value is stored in signed binary format. The sign is assumed to be in the leftmost bit of the leftmost byte. The length of the value must not exceed 256 bytes.

Graphic: Visual representation of a character or symbol.

Halfword: Two adjacent bytes where the left byte is on a halfword boundary.

Halfword Boundary: Even-numbered byte position in core storage, co-incident with the left byte of a halfword.

Inter-Block Gap: A blank space on magnetic tape that separates physical records.

I/O Area: An area (portion) of core storage into which data is read or from which data is written. I/O means Input/Output.

Job Control: A program that is called into core storage to prepare a job to be run. Some of its functions are to assign I/O devices to certain symbolic names, set switches for program use, log (or print) the Job Control statements, and fetch the requested program phase.

Job Control Statement: Any one of the control statements in the input stream that identifies a job or defines its requirements and options.

Label: Can be

1. A physical identification record on magnetic tape located either preceding or following a data file, or both. If a data file extends beyond a single reel of tape, a label can be placed preceding and following the data on each reel.

2. A physical identification record on disk which identifies the volume or file.

Logical Record: A record identified from the standpoint of its content, function, and use, rather than its physical attributes. It is meaningful with respect to the information it contains. (Contrasted with Physical Record.)

Major Control Field: The most significant (in determining the output data set sequence) control field in a logical record.

Merge: The process by which several sequences of logical records are collated to form one sequence. Also a program or routine that performs the process.

Merge Order (Order of Merge): The number of data sets or sequences that are combined during a merging operation.

Merge Pass: The complete process of producing a number of strings on a single output data set by merging strings from each of two or more data sets.

Minor Control Field: The control fields subsidiary to the major control field in the logical record, considered in decreasing order of significance.

Mnemonic: Contraction or abbreviation whose characters are suggestive of the full expression.

Multi-Extent Disk File: File stored on a disk pack in several areas or defined by more than one extent.

Multi-Pack Disk File: File stored on more than one disk pack.

Multi-Reel Tape File: File stored on more than one tape reel.

Multi-Volume Disk File: The same as Multi-Pack Disk File.

Multi-Volume Tape File: The same as Multi-Reel-Tape File.

Operand: Represents the information which must be supplied to define a selective function to the program.

Operation Code: Mnemonic term that identifies the nature of the information in a statement.

Order of Merge: The same as Merge Order.

Packed Decimal: Storage technique whereby two digits or one digit and sign are stored per byte.

Phase: A portion of a program executed as one core-storage load.

Physical Record: A record identified from the standpoint of the manner or form in which it is stored and retrieved; that is, one that is meaningful with respect to access. (Contrasted with Logical Record.)

Primary Tape Drive: See Alternate Tape Drive.

Record: A general term for any unit of data that is distinct from all others when considered in a particular context.

Sequence: A group of logical records whose control fields are in the desired order according to the collating sequence. A high degree of pre-sequencing of the input data set is said to exist if, within each segment, all or most of the respective characters sorted on are within a relatively limited range of numbers, though not necessarily in order.

Single-Extent Disk File: A file stored on disk where the file has exactly one extent.

Single-Pack Disk File: A file completely contained within one disk pack.

Sort: The process by which a data set of logical records is sequenced according to the collating-sequence value of the control field of the records. Also a program that performs the process.

Sort/Merge: A descriptive term meaning "sort or merge". This term is frequently used in connection with a generalized program from which types of sort or merge programs may be defined.

Sort Blocking Factor: The number of logical records in each physical record, as blocked by the Sort/Merge Program for the intermediate storage data sets.

String: Sequenced set of logical records. One string may consist of more than one physical record.

Unpacked-Decimal: Storage technique whereby one digit, with or without sign, is stored per byte.

User-Written Routine: A routine written and supplied by the user and incorporated into the Disk Sort/Merge Program as a modification. Each user-written routine is accessed through a program exit.

Variable-Length Records: Logical records in a data set in which the number of bytes in each record is not a fixed value, but may vary within prescribed limits.

Volume: That portion of a single unit of storage media that is accessible to a single read/write mechanism. For example, a reel of magnetic tape for a 2415 magnetic tape drive, or one 1316 Disk Pack for a 2311 Disk Storage Drive.

Volume Label: Label which uniquely identifies the volume.

Zoned-Decimal: The same as Unpacked-Decimal.

INDEX

READER'S COMMENT FORM

IBM System/360 Model 20
Disk Programming System
Disk Sort/Merge Program

Form C26-3806-1

- How did you use this publication?

As a reference source	☐
As a classroom text	☐
As a self-study text	☐

- Based on your own experience, rate this publication . . .

As a reference source:	Very Good	Good	Fair	Poor	Very Poor
As a text:	Very Good	Good	Fair	Poor	Very Poor

- What is your occupation? ....

- We would appreciate your other comments; please give specific page and line references where appropriate. If you wish a reply, be sure to include your name and address.

- Thank you for your cooperation. No postage necessary if mailed in the U.S.A.

C26-3806-1

YOUR COMMENTS PLEASE . . .

This SRL bulletin is one of a series which serves as reference sources for systems analysts, programmers and operators of IBM systems. Your answers to the questions on the back of this form, together with your comments, will help us produce better publications for your use. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Please note: Requests for copies of publications and for assistance in utilizing your IBM system should be directed to your IBM representative or to the IBM sales office serving your locality.

Fold Fold

FIRST CLASS
PERMIT NO. 1359
WHITE PLAINS, N. Y.

BUSINESS REPLY MAIL
NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY . . .

IBM Corporation
112 East Post Road
White Plains, N. Y. 10601

Attention: Department 813

Fold Fold

IBM

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
[USA Only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]

C26-3806-1

IBM System/360 Printed in U.S.A. C26-3806-1

IBM®

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
[USA Only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]